PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-134032
(43)Date of publication of application : 22.05.1998

(51)Int.Cl. G06F 17/10

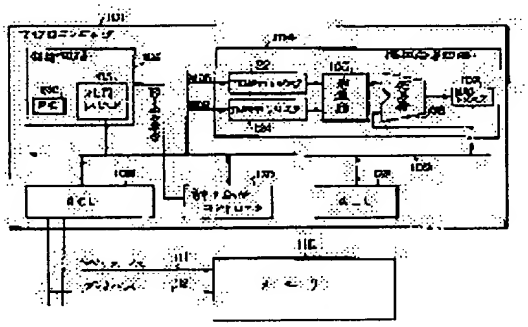
(21)Application number : 09-135922
(22)Date of filing : 08.05.1997

(71)Applicant : SEIKO EPSON CORP
(72)Inventor : KUBOTA SATORU
MIYAYAMA YOSHIYUKI
KUDO MAKOTO

(30)Priority
Priority number : 08234565 Priority date : 04.09.1996 Priority country : JP

(54) INFORMATION PROCESSING CIRCUIT, MICROCOMPUTER, AND ELECTRONIC EQUIPMENT

(57)Abstract:
PROBLEM TO BE SOLVED: To improve memory use efficiency and an execution speed, to eliminate the critical path of product sum operation, and to prevent an overflow by performing product sum operation as many times as specified on the basis of execution frequency information included in a product sum operation instruction.
SOLUTION: A control circuit 102 performs control for receiving instructions including the product sum operation instruction, analyzing the received instructions, and executing the analyzed instructions. The product sum operation instruction contains the execution frequency information specifying the execution frequency of the product sum operation. Then a product sum arithmetic circuit 104 performs the product sum operation as many times as specified with the product sum operation instruction under the control of a control circuit 102. Consequently, the product sum operation can be performed as many times as desired with the one instruction. Therefore, the memory capacity needed for the product sum operation is greatly reduced and the use efficiency of a memory is improved. Further, the need to fetch the product sum operation instruction at each time during the execution of the product sum operation is eliminated to evade the delay of the product sum operation instruction execution.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted r gistration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－134032

(43)公開日　平成10年(1998)５月22日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　17/10		Ｇ０６Ｆ　15/31	Ｓ

審査請求　未請求　請求項の数18　ＦＤ　（全 21 頁）

(21)出願番号　特願平9－135922

(22)出願日　平成９年(1997)５月８日

(31)優先権主張番号　特願平8－234565

(32)優先日　平８(1996)９月４日

(33)優先権主張国　日本（ＪＰ）

(71)出願人　000002369
セイコーエプソン株式会社
東京都新宿区西新宿２丁目４番１号

(72)発明者　久保田　哲
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内

(72)発明者　宮山　芳幸
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内

(72)発明者　工藤　真
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内

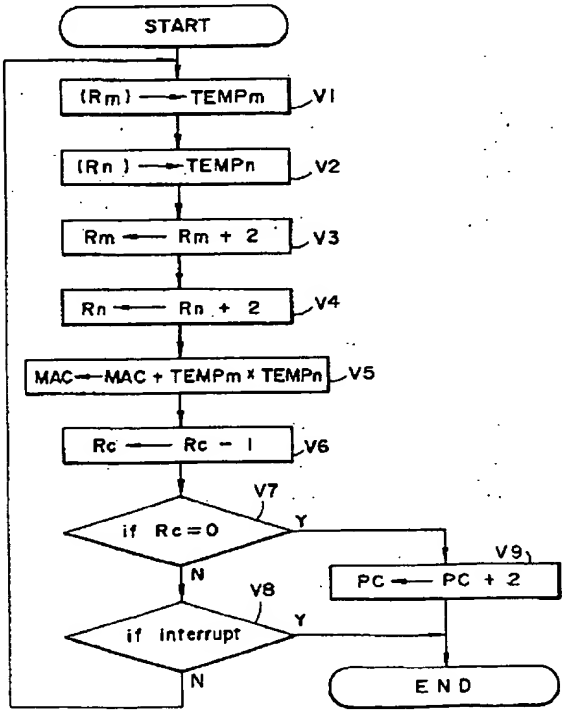
(74)代理人　弁理士　井上　一　（外２名）

(54)【発明の名称】　情報処理回路、マイクロコンピュータ及び電子機器

(57)【要約】

【課題】積和演算命令のメモリ使用効率、実行速度を改善し、積和演算におけるクリティカルパスの改善、オーバーフローの防止を図る。

【解決手段】積和演算回路は、制御回路の制御の下で、積和演算命令に含まれる実行回数情報に基づき特定される回数だけ積和演算を実行する。積和演算の実行回数をレジスタに設定し、積和演算が１サイクル終了する毎にデクリメントし、レジスタの値が０になった時点で積和演算命令を終了する。複数回の積和演算の実行中に割り込みを受け付け、割り込み処理後に積和演算を継続実行する。１回のメモリアクセスで第１、第２の積和入力データを一度に読み出す。１６ビット×１６ビットの乗算結果を３２ビットの加算器で加算し、下位の３２ビットの加算でキャリー、ボローが生じた場合には、次のパイプラインステージで上位の３２ビットのデータをインクリメント又はデクリメントする。



1

【特許請求の範囲】

【請求項１】　積和演算命令を含む命令を受け、該命令を解析し、該命令を実行するための制御を行う制御回路と、

前記積和演算命令に基づき前記制御回路の制御の下で積和演算を実行する積和演算回路とを含み、

前記積和演算回路が、

前記積和演算命令に含まれる実行回数情報に基づき特定される回数だけ積和演算を実行することを特徴とする情報処理回路。

【請求項２】　請求項１において、

前記制御回路が含むレジスタに格納される積和演算の実行回数を、積和演算の実行に同期させてデクリメントする回路を含み、

前記積和演算回路が、

前記実行回数が所与の値になるまで積和演算を実行することを特徴とする情報処理回路。

【請求項３】　請求項１又は２において、

前記積和演算命令が、

積和演算の実行回数用のレジスタ、第１の積和入力データ用のレジスタ及び第２の積和入力データ用のレジスタの中のいずれか１つのレジスタを指定するオペランドを含み、

前記制御回路が、

前記１つのレジスタ以外の他のレジスタを、前記１つのレジスタを指定する前記オペランドから所与のルールにしたがって特定することを特徴とする情報処理回路。

【請求項４】　請求項１又は２において、

前記積和演算命令が、

積和演算の実行回数用のレジスタを指定するオペランドと、第１の積和入力データ用のレジスタを指定するオペランドと、第２の積和入力データ用のレジスタを指定するオペランドとを含むことを特徴とする情報処理回路。

【請求項５】　請求項１又は２において、

前記制御回路が、

積和演算の実行回数に専用のレジスタ、第１の積和入力データに専用のレジスタ及び第２の積和入力データに専用のレジスタの少なくとも１つを含み、

前記積和演算命令が、

前記少なくとも１つの専用のレジスタを暗黙のオペランドとするオペレーションコードを含むことを特徴とする情報処理回路。

【請求項６】　請求項１乃至５のいずれかにおいて、

前記制御回路が、

前記実行回数情報に基づき特定される回数の積和演算の実行中に割り込み要求がなされた場合に該割り込み要求を受け付け、割り込み処理の終了後に、中断された積和演算を継続実行する制御を行うことを特徴とする情報処理回路。

【請求項７】　請求項６において、

2

前記制御回路が、

割り込み要求がなされた場合に、該制御回路が含むプログラムカウンタをインクリメントすることなく積和演算を一旦終了する制御を行い、

割り込み処理の終了後に、積和演算の実行回数用、第１の積和入力データ用及び第２の積和入力データ用のレジスタの割り込み処理分岐時点での内容に基づいて、積和演算を継続実行する制御を行うことを特徴とする情報処理回路。

【請求項８】　請求項６又は７において、

前記実行回数情報に基づき特定される回数だけ積和演算を実行した場合及び前記割り込み要求がなされた場合のいずれかの場合にアクティブになる信号に基づいて、積和演算回路のステートを初期ステートに戻すステートマシーンを含むことを特徴とする情報処理回路。

【請求項９】　請求項６乃至８のいずれかにおいて、

第１の積和入力データ用のレジスタの内容及び第２の積和入力データ用のレジスタの内容が、積和演算の継続実行の際に使用される内容に変化した後に、積和演算回路のステートを初期ステートに戻すステートマシーンを含むことを特徴とする情報処理回路。

【請求項１０】　請求項１乃至９のいずれかにおいて、

前記制御回路が、

第１、第２の積和入力データが隣り合って格納されるメモリ上の領域から、該第１、第２の積和入力データを１回のメモリアクセスで読み出す制御を行うことを特徴とする情報処理回路。

【請求項１１】　積和演算命令を含む命令を受け、該命令を解析し、該命令を実行するための制御を行う制御回路と、

前記積和演算命令に基づき前記制御回路の制御の下で積和演算を実行する積和演算回路とを含み、

前記制御回路が、

第１、第２の積和入力データが隣り合って格納されるメモリ上の領域から、該第１、第２の積和入力データを１回のメモリアクセスで読み出す制御を行うことを特徴とする情報処理回路。

【請求項１２】　請求項１０又は１１において、

前記制御回路と前記メモリとの間でのデータ転送が2^nビットのバスで行われる場合に、転送されるデータの上位2^{n-1}ビットを前記第１の積和入力データとし、下位2^{n-1}ビットを前記第２の積和入力データとすることを特徴とする情報処理回路。

【請求項１３】　請求項１乃至１２のいずれかにおいて、

前記積和演算回路が、

パイプライン処理の第１のステージにおいて、

第１、第２の積和入力データを乗算し、

パイプライン処理の第２のステージにおいて、

所与の第１の積和結果用レジスタに格納されるデータに

前記第１のステージの乗算結果を加算し、
パイプライン処理の第３のステージにおいて、
前記第２のステージの加算でキャリー及びボローのいずれかが生じた場合に、所与の第２の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行うことを特徴とする情報処理回路。

【請求項１４】　積和演算命令を含む命令を受け、該命令を解析し、該命令を実行するための制御を行う制御回路と、
前記積和演算命令に基づき前記制御回路の制御の下で積和演算を実行する積和演算回路とを含み、
前記積和演算回路が、
パイプライン処理の第１のステージにおいて、
第１、第２の積和入力データを乗算し、
パイプライン処理の第２のステージにおいて、
所与の第１の積和結果用レジスタに格納されるデータに前記第１のステージの乗算結果を加算し、
パイプライン処理の第３のステージにおいて、
前記第２のステージの加算でキャリー及びボローのいずれかが生じた場合に、所与の第２の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行うことを特徴とする情報処理回路。

【請求項１５】　請求項１３又は１４において、
前記積和演算回路が、
前記第１、第２の積和結果用レジスタと、
前記第１、第２の積和入力データを乗算する乗算器と、
前記第１の積和結果用レジスタに格納されるデータに前記乗算器からの乗算結果を加算する加算器と、
前記加算器からのキャリー信号、ボロー信号に基づいて、前記第２の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行う回路とを含むことを特徴とする情報処理回路。

【請求項１６】　請求項１３乃至１５のいずれかにおいて、
前記第１、第２の積和入力データの各々が2^{n-1}ビットのデータであり、前記第１、第２の積和結果用レジスタの各々が2^{n}ビットのレジスタであることを特徴とする情報処理回路。

【請求項１７】　半導体基板上に集積されたマイクロコンピュータであって、
請求項１乃至１６のいずれかの情報処理回路と、
バスコントロール回路、メモリ、割り込みコントローラ、タイマ回路、アナログインターフェース回路、データ転送制御回路及びＩ／Ｏ回路の少なくとも１つとを含むことを特徴とするマイクロコンピュータ。

【請求項１８】　請求項１７のマイクロコンピュータと、
前記マイクロコンピュータの処理対象となるデータの入力源と、
前記マイクロコンピュータにより処理されたデータを出力するための出力装置とを含むことを特徴とする電子機器。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、情報処理回路、マイクロコンピュータ及び電子機器に関する。

【０００２】

【背景技術及び発明が解決しようとする課題】近年、積和演算命令を高速に実行できるマイクロコンピュータに対する需要が高まっている。高速な積和演算が可能になると、これまでＤＳＰ（Digital Signal Processor）、画像処理専用ＩＣ、音処理専用ＩＣが行っていた処理をマイクロコンピュータに代行させることが可能となり、製品の低コスト化、システムの簡素化を図れるからである。

【０００３】マイクロコンピュータにおける、積和演算命令は例えば次のように実行される。まずメモリ上の第１の領域に第１の積和入力データを格納しておくと共に第２の領域に第２の積和入力データを格納しておく。次にマイクロコンピュータが内蔵する汎用レジスタの内容で指定される２個のアドレスを用いて、上記第１、第２の領域に格納された第１、第２の積和入力データをメモリから読み出す。そしてこれらの第１、第２の積和入力データを乗算し、その結果を積和演算回路が内蔵する積和結果用のレジスタ（ＭＡＣレジスタ）に加算する動作を行う。

【０００４】しかしながら、積和演算命令の実行が可能な上記マイクロコンピュータには次のような課題がある。

【０００５】（１）積和演算を複数回実行しようとした場合、積和演算命令を繰り返す回数分だけ、積和演算の命令を並べたプログラムを作成する必要がある。そのため、積和演算の実行回数が増大すると、それに応じて積和演算命令を格納するのに必要なメモリ容量が増加してしまう。これを解決する１つの手法として、積和演算の実行回数をデクリメントしながら積和演算を実行し、実行回数が零になった時にループを抜けるようなプログラムを作成する手法も考えられる。しかしながら、この手法によると、積和演算の１回当たりの実行に要する時間が長くなる。

【０００６】（２）積和演算を連続して実行した場合、１回当たりの実行時間が、第１、第２の積和入力データをメモリから読み出す時間によって制限されてしまう。

【０００７】（３）これまでは例えば各々が１６ビットの第１、第２の積和入力データを乗算し、乗算結果を４８ビットのＭＡＣレジスタ（積和結果用レジスタ）に加算していた。この場合、４８ビットの加算を１クロック内で完了する必要があり、この加算処理がクリティカルパスとなる。またＭＡＣレジスタが４８ビットのビット長しか持たないため、積和演算の実行回数が多くなる

と、すぐにオーバーフローしてしまう。

【０００８】本発明は、以上のような技術的課題に鑑みてなされたものであり、その目的とするところは、積和演算命令を用いるプログラムのメモリ使用効率の向上を図れる情報処理回路、マイクロコンピュータ及び電子機器を提供することにある。

【０００９】また本発明の他の目的は、積和演算命令の実行速度を改善できる情報処理回路等を提供することにある。

【００１０】また本発明の他の目的は、積和演算におけるクリティカルパスの解消、積和演算におけるオーバーフローの防止を図れる情報処理回路等を提供することにある。

【００１１】

【課題を解決するための手段】上記課題を解決するために本発明に係る情報処理回路は、積和演算命令を含む命令を受け、該命令を解析し、該命令を実行するための制御を行う制御回路と、前記積和演算命令に基づき前記制御回路の制御の下で積和演算を実行する積和演算回路とを含み、前記積和演算回路が、前記積和演算命令に含まれる実行回数情報に基づき特定される回数だけ積和演算を実行することを特徴とする。

【００１２】本発明によれば、積和演算命令に、積和演算の実行回数を特定するための実行回数情報が含まれる。そして積和演算回路は、制御回路の制御の下で、積和演算命令により特定された回数の積和演算を行う。これにより、１回の命令で所望の回数の積和演算を実行することが可能となる。従って、積和演算の回数だけ積和演算命令を並べる手法に比べて、積和演算に必要とされるメモリ容量を大幅に削減でき、メモリの使用効率を向上できる。また積和演算を実行中に積和演算命令を毎回フェッチする必要がなくなり、積和演算命令実行の遅延を回避できる。

【００１３】また本発明は、前記制御回路が含むレジスタに格納される積和演算の実行回数を、積和演算の実行に同期させてデクリメントする回路を含み、前記積和演算回路が、前記実行回数が所与の値になるまで積和演算を実行することを特徴とする。このようにすれば、積和演算の実行毎に実行回数をメモリから読み出す必要がなくなり、処理速度を向上できる。またこのようにすることで、複数回の積和演算の実行中に割り込みが発生した場合に、割り込み処理後に、レジスタに格納されている実行回数に基づいて積和演算を継続実行することが可能となる。

【００１４】なお本発明では、前記積和演算命令が、積和演算の実行回数用のレジスタ、第１の積和入力データ用のレジスタ及び第２の積和入力データ用のレジスタの中のいずれか１つのレジスタを指定するオペランドを含み、前記制御回路が、前記１つのレジスタ以外の他のレジスタを、前記１つのレジスタを指定する前記オペランドから所与のルールにしたがって特定することが望ましい。このようにすれば、命令のビット長を小さくすることができ、プログラムのコードサイズの縮小化を図れる。

【００１５】なお前記積和演算命令が、積和演算の実行回数用のレジスタを指定するオペランドと、第１の積和入力データ用のレジスタを指定するオペランドと、第２の積和入力データ用のレジスタを指定するオペランドとを含むようにしてもよい。また前記制御回路が、積和演算の実行回数に専用のレジスタ、第１の積和入力データに専用のレジスタ及び第２の積和入力データに専用のレジスタの少なくとも１つを含み、前記積和演算命令が、前記少なくとも１つの専用のレジスタを暗黙のオペランドとするオペレーションコードを含むようにしてもよい。

【００１６】また本発明は、前記制御回路が、前記実行回数情報に基づき特定される回数の積和演算の実行中に割り込み要求がなされた場合に該割り込み要求を受け付け、割り込み処理の終了後に、中断された積和演算を継続実行する制御を行うことを特徴とする。このようにすれば、複数回の積和演算の実行の際に、割り込みが長時間待たされるという問題を解消できる。

【００１７】また本発明は、前記制御回路が、割り込み要求がなされた場合に、該制御回路が含むプログラムカウンタをインクリメントすることなく積和演算を一旦終了する制御を行い、割り込み処理の終了後に、積和演算の実行回数用、第１の積和入力データ用及び第２の積和入力データ用のレジスタの割り込み処理分岐時点での内容に基づいて、積和演算を継続実行する制御を行うことを特徴とする。このようにすれば、割り込み処理の終了後に、適正な第１、第２の積和入力データを用いて、適正な実行回数だけ積和演算を継続実行することが可能となる。

【００１８】また本発明は、前記実行回数情報に基づき特定される回数だけ積和演算を実行した場合及び前記割り込み要求がなされた場合のいずれかの場合にアクティブになる信号に基づいて、積和演算回路のステートを初期ステートに戻すステートマシーンを含むことを特徴とする。このようにすれば、ステートマシーンの構成の簡素化を図れる。

【００１９】また本発明は、第１の積和入力データ用のレジスタの内容及び第２の積和入力データ用のレジスタの内容が、積和演算の継続実行の際に使用される内容に変化した後に、積和演算回路のステートを初期ステートに戻すステートマシーンを含むことを特徴とする。このようにすれば、割り込み処理の終了後に、適正な第１、第２の積和入力データを用いて積和演算を継続実行することが可能となる。

【００２０】また本発明は、前記制御回路が、第１、第２の積和入力データが隣り合って格納されるメモリ上の

領域から、該第１、第２の積和入力データを１回のメモリアクセスで読み出す制御を行うことを特徴とする。これまでは、積和演算の１回当たりの実行時間は、２回のメモリアクセスに要する時間に制限されていたが、本発明によれば、上記実行時間を、１回のメモリアクセスに要する時間にまで短縮できる

また本発明は、前記制御回路と前記メモリとの間でのデータ転送が2^nビットのバスで行われる場合に、転送されるデータの上位2^{n-1}ビットを前記第１の積和入力データとし、下位2^{n-1}ビットを前記第２の積和入力データとすることを特徴とする。このようにすることで、第１、第２の積和入力データを2^nビットのバスを用いて１回のメモリアクセスで読み込むことが可能となる。

【００２１】また本発明は、前記積和演算回路が、パイプライン処理の第１のステージにおいて、第１、第２の積和入力データを乗算し、パイプライン処理の第２のステージにおいて、所与の第１の積和結果用レジスタに格納されるデータに前記第１のステージの乗算結果を加算し、パイプライン処理の第３のステージにおいて、前記第２のステージの加算でキャリー及びボローのいずれかが生じた場合に、所与の第２の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行うことを特徴とする。このようにすれば、乗算結果を加算する際に生じるクリティカルパスの問題、オーバーフローの問題を解消できる。

【００２２】なお本発明は、前記積和演算回路が、前記第１、第２の積和結果用レジスタと、前記第１、第２の積和入力データを乗算する乗算器と、前記第１の積和結果用レジスタに格納されるデータに前記乗算器からの乗算結果を加算する加算器と、前記加算器からのキャリー信号、ボロー信号に基づいて、前記第２の積和結果用レジスタに格納されるデータのインクリメント及びデクリメントのいずれかを行う回路とを含むことが望ましい。

【００２３】また本発明は、前記第１、第２の積和入力データの各々が2^{n-1}ビットのデータであり、前記第１、第２の積和結果用レジスタの各々が2^nビットのレジスタであることを特徴とする。このようにすれば、2^nビットの第１の積和結果用レジスタと2^nビットの第２の積和結果用レジスタとを合わせて2^{n+1}ビットのレジスタに対して積和演算結果を加算できるため、積和演算の実行可能回数を、事実上、無限大にすることが可能となる。

【００２４】また本発明は、半導体基板上に集積されたマイクロコンピュータであって、上記のいずれかの情報処理回路と、バスコントロール回路、メモリ、割り込みコントローラ、タイマ回路、アナログインターフェース回路、データ転送制御回路及びＩ／Ｏ回路の少なくとも１つとを含むことを特徴とする。このようなマイクロコンピュータを用いることで、これまでＤＳＰ、画像処理専用ＩＣ或いは音処理専用ＩＣが行っていた処理を代行することが可能となる。

【００２５】また本発明に係る電子機器は、上記のマイクロコンピュータと、前記マイクロコンピュータの処理対象となるデータの入力源と、前記マイクロコンピュータにより処理されたデータを出力するための出力装置とを含むことを特徴とする。このようにすれば、これまでＤＳＰ等が行っていた画像処理、音処理をマイクロコンピュータが代行することが可能となり、電子機器の低コスト化、コンパクト化、低消費電力化を図ることが可能となる。

【００２６】

【発明の実施の形態】以下、本発明の好適な実施形態について図面を用いて詳細に説明する。なお以下では、本発明の情報処理回路をマイクロコンピュータに適用した場合を主に例にとり説明を行う。

【００２７】（実施例１）実施例１は、積和演算命令に含まれる実行回数情報に基づき特定される回数だけ積和演算回路に積和演算を実行させる実施例である。

【００２８】図１に、積和演算回路１０４を内蔵するマイクロコンピュータ１０１のブロック図を示す。この機能ブロック図は、実施例１及び後述の実施例２、３、４に共通のブロック図である。図１では、マイクロコンピュータ１０１は３２ビットのデータを処理する。また積和演算回路１０４は、各々が１６ビットの第１、第２の積和入力データＭＤＡ、ＭＤＢを乗算し、その乗算結果である３２ビットのデータを６４ビットのＭＡＣレジスタ１０７に加算する。但し本発明の適用範囲は、マイクロコンピュータ及び積和演算回路の処理するデータのビット長、もしくはマイクロコンピュータの内蔵する汎用レジスタの数等により限定されるものではない。

【００２９】図１のマイクロコンピュータ１０１は、３２ビットのデータを処理する制御回路１０２と、マイクロコンピュータ１０１とメモリ１１０とを接続するバスを制御するバスコントロールユニットＢＣＵ１０８と、積和演算を実行する積和演算回路１０４と、マイクロコンピュータの内部や外部からの種々の割り込みを受け付け制御回路１０２に割り込み要求を出す割り込みコントローラ１３０と、データの加減演算などの算術演算及び論理和、論理積、論理シフトなどの論理演算を行うＡＬＵ（算術論理演算ユニット）１３２を含む。

【００３０】ここで制御回路１０２は、積和演算命令を含む命令を受け、受け付けた命令を解析し、解析した命令を実行するための制御を行うものであり、１６ビット長の命令を使用している。また制御回路１０２は、Ｒ０からＲ１５までの１６本の３２ビットのレジスタから成る汎用レジスタ１０３と、ＰＣ（プログラムカウンタ）１２０とを含む。この制御回路１０２の制御の下で、積和演算回路１０４は積和演算を実行し、ＡＬＵ１３２は算術演算、論理演算を実行する。そしてこれらの制御回路１０２、積和演算回路１０４及びＡＬＵ１３２が、マ

イクロコンピュータ１０１の中央処理ユニット（ＣＰＵ）として機能する。

【００３１】制御回路１０２、ＢＣＵ１０８及び積和演算回路１０４は、内部データバス１０９を介してデータ転送を行う。ＢＣＵ１０８は、外部アドレスバス１１１と外部データバス１１２を用いて、メモリ１１０から第１、第２の積和入力データＭＤＡ、ＭＤＢを読み込む。但し本発明の適用範囲は、メモリ１１０がマイクロコンピュータ１０１に内蔵されるか、マイクロコンピュータ１０１の外部に設けられるかによって限定されるものではない。

【００３２】積和演算回路１０４は、第１、第２の積和入力データＭＤＡ、ＭＤＢを一時的に保持するＴＥＭＰｍレジスタ１２２及びＴＥＭＰｎレジスタ１２４と、一時的に保持されたＭＤＡ、ＭＤＢを乗算する乗算器１０５と、この乗算結果を用いて加算処理を行う加算器１０６と、加算結果を保持する６４ビットのＭＡＣレジスタ（積和結果用レジスタ）１０７を含む。積和演算回路１０４は、１６ビットのＭＤＡ、ＭＤＢを入力として、その乗算結果とＭＡＣレジスタ１０７の内容とを加算し、その加算結果をＭＡＣレジスタ１０７に格納する動作を行う。

【００３３】次に図２のフローチャート及び図３（Ａ）、（Ｂ）、（Ｃ）を用いて本実施例の動作について説明する。

【００３４】まず積和演算命令実行前に、図３（Ａ）に示すように、メモリ上の第１の領域１０に第１の積和入力データＭＤＡ0～ＭＤＡlを格納し、第２のメモリ領域１２に第２の積和入力データＭＤＢ0～ＭＤＢlを格納しておく。また汎用レジスタ１０３が含むレジスタＲｍ、Ｒｎに第１、第２の領域１０、１２の先頭アドレスをロードしておき、第１、第２の積和入力データの先頭データＭＤＡ0、ＭＤＢ0をＲｍ、Ｒｎが指すようにしておく。また汎用レジスタ１０３が含むレジスタＲｃに、積和演算の実行回数をロードしておくと共に、ＭＡＣレジスタ１０７を初期化しておく。

【００３５】この状態で制御回路１０２が積和演算命令を受け付けると（即ち図３（Ａ）に示すようにＰＣ１２０の指す命令が積和演算命令であると）、この積和演算命令を解析する制御回路１０２の制御の下で、積和演算命令実行のための種々の処理が行われる。

【００３６】即ちまず、レジスタＲｍに格納されるアドレスにより指定される第１の積和入力データＭＤＡ0を、ＢＣＵ１０８を介してメモリ１１０から読み出し、ＴＥＭＰｍレジスタ１２２に格納する（図２のステップＳ１）。同様にＲｎが指すＭＤＢ0を、ＢＣＵ１０８を介してメモリ１１０から読み出し、ＴＥＭＰｎレジスタ１２４に格納する（ステップＳ２）。なお、ステップＳ１、Ｓ２の（Ｒｍ）、（Ｒｎ）は、各々、Ｒｍ、Ｒｎに格納されるアドレスにより指定されるメモリ上のデータ（第１、第２の積和入力データ）を意味する。

【００３７】次に、Ｒｍ、Ｒｎに格納されるアドレスを＋２（番地）する（ステップＳ３、Ｓ４）。本実施例では、メモリ空間は、８ビットの１バイトを最小単位としてアドレッシングされており、第１、第２の積和入力データは１６ビットのデータである。従ってＲｍ、Ｒｎのアドレスを＋２すると、図３（Ｂ）に示すようにＲｍ、Ｒｎは次の積和入力データＭＤＡ1、ＭＤＢ1を指すようになる。

【００３８】次にＴＥＭＰｍレジスタ１２２、ＴＥＭＰｎレジスタ１２４に格納されている１６ビットのＭＤＡ0、ＭＤＢ0を乗算器１０５を用いて乗算し、その乗算結果とＭＡＣレジスタ１０７の内容を加算器１０６により加算し、その加算結果をＭＡＣレジスタ１０７に格納する（ステップＳ５）。

【００３９】次に、レジスタＲｃに格納される積和演算の実行回数をデクリメントする（ステップＳ６）。即ち積和演算の実行回数を、積和演算に同期させてデクリメントする。なお図２では積和演算が終了する毎に実行回数をデクリメントしているが、実行回数のデクリメントは少なくとも積和演算の実行に同期するものであればよい。このデクリメント処理は、例えば図１のＡＬＵ１３２が行う。

【００４０】次に、Ｒｃに格納される積和演算の実行回数が所与の値、例えば零か否かを判断する（ステップＳ７）。零でない場合には、ステップＳ１に戻る。この時、前述のようにステップＳ３、Ｓ４でＲｍ、Ｒｎが＋２（番地）されているため、積和演算のオペランドは次の積和入力データＭＤＡ1、ＭＤＢ１となっている（図３（Ｂ）参照）。

【００４１】一方、Ｒｃに設定された実行回数だけ積和演算が繰り返され、Ｒｃの値が零になると、図３（Ｃ）に示すようにＰＣ１２０に格納されるアドレスを＋２（番地）する（ステップＳ８）。これにより積和演算命令が終了する共にＰＣ１２０が次の命令を指すようになる。本実施例では、命令長は２バイトの１６ビットであるため、ＰＣ１２０の値を＋２（番地）すると、２バイト先の次の命令が指されることになる。

【００４２】以上のような処理が行われるように制御回路１０２が、積和演算回路１０４、ＢＣＵ１０８、ＡＬＵ１３２などを制御することで、１つの命令で所望の回数だけ積和演算を実行することが可能となる。

【００４３】図４に、本実施例の比較例となるマイクロコンピュータの処理のフロチャートを示す。図４のステップＴ１～Ｔ５は図２のステップＳ１～Ｓ５と同様である。しかしながら、図４の比較例では図２の本実施例と異なり、Ｒｃの値をデクリメントしたり、Ｒｃの値が零か否かを判断することなく、ＰＣ１２０の値を＋２して積和演算命令を終了する。また積和演算命令に、積和演算の実行回数を特定するための情報が含まれていない。

【００４４】従って図４の手法で積和演算を所望の回数だけ実行するためには、図５（Ａ）に示すように、その回数分だけ積和演算の命令を並べたプログラムを作成する必要があり、これはプログラムのコードサイズの増大化、命令を格納するメモリの大規模化等の問題を招く。

【００４５】このような問題を解決する１つの手法として、図５（Ｂ）に示すように、積和演算を実行する回数Ｃをデクリメントしながら積和演算を実行し（ステップＵ１、Ｕ２）、実行回数Ｃが零になった場合にループを抜ける（ステップＵ３）ようにプログラムする手法も考えられる。しかしながら、この手法によると、積和演算の１回当たりの実行に要する時間が長くなる。即ち、この手法によると、積和演算命令（ステップＵ１）の実行に要する時間以外に、実行回数Ｃをデクリメントする命令（ステップＵ２）に要する時間、実行回数が零か否かを判断する命令（ステップＵ３）に要する時間が余計に必要になる。従って、図２の本実施例に比べ、最低でも２クロック分だけ処理時間が増えてしまい、処理速度の低下を招く。

【００４６】これに対して本実施例では、実行回数分だけ積和演算命令を並べるプログラムや、実行回数をデクリメントし実行回数が零か否かを判断するプログラムを作成する必要がないため、コードサイズの効率化、命令を格納するメモリの小規模化、処理の高速化を図りながら、１つの積和演算命令で所望の回数だけ積和演算命令を実行することが可能となる。

【００４７】また本実施例によれば、積和演算を実行中に積和演算命令を毎回フェッチする必要がない。従って、積和入力データの読み込みと積和演算命令のフェッチが競合することで発生する積和演算命令実行の遅延を回避できると供に、無駄なフェッチによる電力消費を節減できる。

【００４８】さて積和演算の実行回数を特定させる積和演算命令の実施形態としては種々のものが考えられる。

【００４９】例えば図６（Ａ）では、積和演算命令に、複数の命令の中から積和演算命令を指定するための６ビットのオペレーションコードと、レジスタＲｃを１６個の汎用レジスタの中から指定する４ビットのオペランドとを含ませている。そしてこの場合には、制御回路１０２が、第１の積和入力データ用のレジスタＲｍ及び第２の積和入力データ用のレジスタＲｎを、上記Ｒｃを指定するオペランドに基づき所与のルールにしたがって特定する。例えば図６（Ａ）の積和演算命令で汎用レジスタＲ１３がＲｃに割り当てられた場合には、Ｒ１３を＋１、＋２した汎用レジスタＲ１４、Ｒ１５に、各々、Ｒｍ、Ｒｎを割り当てる。このようにすることで、命令長を１６ビット以内に抑えることが可能となり、コードサイズの効率化、命令を格納するメモリの小規模化を図れる。特に全ての命令の長さを例えば１６ビットの固定長にしてコードサイズの効率化を図る命令セットのアーキテクチャにおいては、図６（Ａ）の手法は有効である。なお図６（Ａ）では、積和演算命令が、Ｒｃを指定するオペランドを含んでいるが、Ｒｍ或いはＲｎを指定するオペランドを含ませるようにしてもよい。

【００５０】また図６（Ｂ）では、積和演算命令に、６ビットのオペレーションコード、Ｒｃを指定する４ビットのオペランド、Ｒｍを指定する４ビットのオペランド及びＲｎを指定する４ビットのオペランドを含ませている。即ち積和演算命令によりＲｃ、Ｒｍ、Ｒｎを直接指定している。この手法は図６（Ａ）に比べて、命令長が長くなるという不利点があるが、１つのオペランドから他のオペランドを特定する処理が必要がないという利点がある。

【００５１】また図６（Ｃ）では、Ｒｃ、Ｒｍ、Ｒｎを、各々、実行回数、第１、第２の積和入力データに専用のレジスタにしている。この場合には、積和演算命令は、これらの専用のレジスタを暗黙のオペランドとするオペレーションコードを含むことになる。このようにすることでコードサイズの効率化を図れる。なお図６（Ｃ）では、Ｒｃ、Ｒｍ、Ｒｎを全て専用レジスタにしているが、これらの中の１つ或いは２つのみを専用レジスタにするようにしてもよい。

【００５２】なお本実施例では、第１、第２の積和入力データについてはレジスタＲｍ、Ｒｎに格納されるアドレスに基づいて積和演算の実行毎にメモリから読み出しているのに対して、実行回数についてはＲｃ自体に格納されているものを使用している。積和演算の実行毎に実行回数をメモリから読み出すと、処理速度の低下を招くからである。また本実施例では、実行回数を、一度メモリーからレジスタＲｃにロードした後にＡＬＵ１３２を用いてデクリメントしているため、積和入力データの場合と異なり再度メモリから実行回数を読み出す必要性がないからである。但し本発明の範囲は、このような実行回数の指定手法に限定されるものではない。

【００５３】（実施例２）実施例２は、複数回の積和演算の実行中に割り込みを受け付けると共に、割り込み終了後に、中断された積和演算を継続実行する実施例である。以下、図７のフローチャート及び図８（Ａ）、（Ｂ）を用いて実施例２について説明する。

【００５４】前述の実施例１では、１つの積和演算命令で複数回の積和演算を実行することができる。しかしながら、この複数回の積和演算の実行は１つの命令と見なされるため、Ｒｃに設定された実行回数分の積和演算が完了しない限り、ＰＣ１２０の値は変化せず、次の命令に移行しない。一方、図１の割り込みコントローラ１３０からの割り込み要求は、通常は、命令と命令の境界で処理される。従って複数回の積和演算を実行している期間、割り込み処理への移行が長時間待たされるという不具合が生じる。

【００５５】そこで実施例２では、図７に示すような処

理を行っている（なお図７のステップＶ１～Ｖ６は図２のステップＳ１～Ｓ６と同様である）。即ちＲｃの値である積和演算の実行回数が零か否かの判断（ステップＶ７）の後に、割り込み要求が存在するか否かを判断する（ステップＶ８）。そして割り込み要求が存在しない場合は、積和演算の処理を継続し、次の積和入力データに基づく積和演算を行う。一方、割り込み要求が存在する場合は、ＰＣ１２０をインクリメントすることなく（ステップＶ９の処理を省略する）、積和演算命令を一旦終了する。図８（Ａ）では、例えば積和入力データＭＤＡK-1、ＭＤＢK-1に対する積和演算処理の際に割り込み要求が生じている。この場合、図７のステップＶ３、Ｖ４の処理により、Ｒｍ、Ｒｎは次の積和入力データＭＤＡK、ＭＤＢKを指すようになっている。またＰＣ１２０は、次の命令を指さず積和演算命令を指したままとなっている（図８（Ａ）のＨ１参照）。

【００５６】積和演算命令を終了することで、割り込み処理プログラムに分岐して、割り込み処理を実行することが可能となる。割り込み処理は、通常、命令と命令の境界で行われるからである。ここで割り込み処理のハンドラは、割り込み処理に分岐する前に、割り込み処理が終了した時の戻りアドレスを指すＰＣ１２０の値をスタックする。しかしながら図７のステップＶ８、Ｖ９に示すように本実施例では、積和演算命令の実行中に割り込み処理に分岐した場合には、ＰＣ１２０の値を＋２することなく積和演算命令を終了している。従って、図８（Ｂ）のＨ２に示すようにＰＣ１２０は積和演算命令を指したままとなっているため、割り込み処理の終了後に、再度同じ積和演算命令が実行されることになる。

【００５７】そして、この時、図８（Ｂ）のＨ３、Ｈ４、Ｈ５に示すように、Ｒｃ、Ｒｍ及びＲｎの値は、割り込み処理に分岐する時点での値になっている。従って、割り込み処理で中断された積和演算を適正に継続実行することが可能となる。即ち、割り込み処理による中断前に、積和入力データＭＤＡK-1、ＭＤＢK-1までの処理が完了しており、中断後に、ＭＤＡK、ＭＤＢKに対する処理から積和演算を再実行できるようになる。

【００５８】以上説明した実施例２によれば、複数回の積和演算の実行中に割り込みを受け付け割り込み処理を行うことができ、しかも、割り込み処理の待ち時間も、図４の場合の待ち時間と同等にすることができる。

【００５９】（実施例３）実施例３は、第１、第２の積和入力データが隣り合って格納されるメモリ上の領域から、これらの第１、第２の積和入力データを１回のメモリアクセスで読み出す実施例である。以下、図９のフローチャート及び図１０（Ａ）、（Ｂ）、（Ｃ）を用いて実施例３について説明する。

【００６０】実施例３では、図１０（Ａ）に示すように、第１の積和入力データＭＤＡと第２の積和入力データＭＤＢとを隣り合わせてメモリ上に格納する。例えばＭＤＡ0の次にはＭＤＢ0が格納され、ＭＤＢ0の次にはＭＤＡ1、ＭＤＢ1が格納される。即ちＮを自然数として、第１の積和入力データＭＤＡは４Ｎ番地のアドレスに、第２の積和入力データＭＤＢは４Ｎ＋２番地のアドレスに格納される。この点において、第１の領域１０に第１の積和入力データＭＤＡがひとまとめに格納され、第２の領域１２に第２の積和入力データＭＤＢがひとまとめに格納される図３（Ａ）、（Ｂ）、（Ｃ）と相違する。

【００６１】そして実施例３では、実施例１、２と異なり、３個ではなく２個のレジスタＲｍ、Ｒｃを使用する。即ちＲｍによりＭＤＡ0、ＭＤＢ0を読み出し、Ｒｍの値を＋４することによりＭＤＡ1、ＭＤＢ1を読み出すようにする。

【００６２】実施例３の動作について説明する。まずレジスタＲｍに格納されるアドレスにより指定される第１、第２の積和入力データＭＤＡ0、ＭＤＢ0をメモリから読み出し、読み出されたデータの上位１６ビット（２バイト）をＴＥＭＰｍレジスタ１２２に、下位１６ビットをＴＥＭＰｎレジスタ１２４に格納する（ステップＷ１）。

【００６３】即ち本実施例では、図１０（Ａ）に示すように、メモリ１１０と積和演算回路１０４の間のデータ転送は３２ビットのバスにより行われる。従って、１回のメモリアクセスで、３２ビット（４バイト）のデータを読み込むことができ、その読み出しデータの上位１６ビットが第１の積和入力データＭＤＡ0となり、下位１６ビットが第２の積和入力データＭＤＢ0になる。

【００６４】積和入力データを読み出した後、Ｒｍの値を＋４し、積和演算を実行する（ステップＷ２、Ｗ３）。Ｒｍの値を＋４することにより、図１０（Ａ）に示すように次の積和入力データＭＤＡ1、ＭＤＢ1を指定することが可能となる。なおステップＷ６～Ｗ９の処理は、図７のステップＶ６～Ｖ９と同様である。

【００６５】実施例１、２では、図１０（Ｂ）に示すように、１回のメモリアクセス（１クロック期間）で、第１、第２の積和入力データＭＤＡ、ＭＤＢのいずれか一方しか読み出せない。そして積和演算はＭＤＡ、ＭＤＢの両方が揃わないと実行できないため、結局、積和演算を２クロック毎にしか実行できない。即ち積和演算の１回当たりの実行時間は、２回のメモリアクセス分の時間となっていた。

【００６６】これに対して実施例３では、図１０（Ｃ）に示すように、１回のメモリアクセスで第１、第２の積和入力データＭＤＡ、ＭＤＢの両方を読み出すことができる。従って、積和演算を１クロック毎に行うことが可能となり、積和演算の実行時間を１回のメモリアクセス分の時間とすることが可能となる。これにより、処理速度を格段に向上できる。

【００６７】（実施例４）実施例４は、パイプライン方

式の積和演算において、第１のステージで乗算を行い、第２のステージで乗算結果を下位の第１の積和結果用レジスタに加算し、第１の積和結果用レジスタがオーバーフローした場合に、第３のステージで上位の第２の積和結果用レジスタをインクリメント又はデクリメントする実施例である。

【００６８】図１１（Ａ）に、実施例４の積和演算回路のブロック図を示す。この積和演算回路は、乗算器１０５、加算器１０６-1、インクリメント・デクリメンタ１０６-2、ＭＡＣレジスタ（積和結果用レジスタ）の下位のレジスタであるＡＬＲ１０７-1、ＭＡＣレジスタの上位のレジスタであるＡＨＲ１０７-2を含む。

【００６９】図１１（Ｂ）に示すように、パイプライン処理の第１のステージでは、乗算器１０５が第１、第２の積和入力データＭＤＡ、ＭＤＢを乗算する。次にパイプライン処理の第２のステージにおいて、加算器１０６-1が、ＡＬＲ１０７-1（第１の積和結果用レジスタ）に格納されるデータに第１のステージの乗算結果を加算する。そして第２のステージの加算で正のオーバーフローが生じキャリー信号がアクティブになった場合には、パイプライン処理の第３のステージにおいて、インクリメント・デクリメンタ１０６-2がＡＨＲ１０７-2（第２の積和結果用レジスタ）に格納されるデータをインクリメントする。一方、第２のステージの加算で負のオーバーフローが生じ、ボロー信号がアクティブになった場合には、第３のステージにおいて、インクリメント・デクリメンタ１０６-2がＡＨＲ１０７-2に格納されるデータをデクリメントする。

【００７０】このように実施例４では、１６ビット×１６ビットの乗算で得られた結果を、ＡＬＲ１０７-1、ＡＨＲ１０７-2から成る６４ビットのＭＡＣレジスタに加算する。そしてこの加算を、下位の３２ビットの加算と上位の３２ビットの加算に分け、下位の３２ビットの加算をパイプライン処理の第２のステージで実行し、上位の３２ビットの加算（インクリメント又はデクリメント）をパイプライン処理の第３のステージで実行する。これにより以下の効果を得ることができる。

【００７１】　加算器１０６-1を４８ビットではなく３２ビットにすることができるため、４８ビットの加算器を使用する場合に問題となっていたクリティカルパスを解消できる。

【００７２】　ＭＡＣレジスタ（ＡＬＲ１０７-1、ＡＨＲ１０７-2）のビット数を６４ビットに拡張できるため、積和演算の際のオーバーフロー（飽和）の可能性を減らすことができると共に、積和演算の実行回数を2^{32}－１回にすることができ、事実上、無制限にすることができる。特に実施例４は実施例１との組み合わせにおいて特有の効果を奏する。即ち積和演算命令に実行回数を特定する情報を含ませる実施例１の手法によれば、積和演算命令を実行回数分だけ並べたプログラムを作成する必要がないため、ユーザが指定する積和演算の実行回数が非常に大きくなる可能性がある。実施例４によれば、積和演算命令の実行回数を事実上無制限にできるため、このような大きな実行回数の指定に対しても対処できる。

【００７３】　通常の加算器に比べてハードウェア規模が小さいインクリメンタ・デクリメンタ１０６-2を用いて上位３２ビットの加算処理ができる。従ってＭＡＣレジスタ（ＡＬＲ１０７-1、ＡＨＲ１０７-2）のビット数を６４ビットに拡張したにも関わらず、ハードウェアの大規模化を最小限に抑えることができる。

【００７４】なお１つのハードウェアを用いて符号付きのデータの乗算と符号なしのデータの乗算を扱えるようにするためには、乗算器１０５を１７ビット×１７ビットの構成とすることが望ましい。また図１１（Ａ）のキャリー信号は、加算器１０６-1がキャリーを発生し且つその時のデータが正の場合にアクティブになる。一方、ボロー信号は、加算器１０６-1がキャリーを発生し且つその時のデータが負の場合にアクティブになる。また符号無しのデータのみを扱う場合には、ボロー信号は必要なく、インクリメント・デクリメンタ１０６-2はインクリメンタの機能を有するのみでよい。

【００７５】（実施例５）実施例５は、図１の制御回路１０２、積和演算回路１０４、ＡＬＵ１３２の詳細例に関する実施例であり、図１２にそのブロック図を示す。

【００７６】図１２において、Ｉ＿ＡＤＤＲ＿ＢＵＳは命令アドレスバスであり、Ｉ＿ＤＡＴＡ＿ＢＵＳは命令データバスである。これらのバスを用いて命令メモリ１１０-1から積和演算命令などの命令が読み出される。またＤ＿ＡＤＤＲ＿ＢＵＳはデータアドレスバス、Ｄ＿ＤＡＴＡ＿ＢＵＳはデータバスであり、これらのバスを用いてデータメモリ１１０-2から第１、第２の積和入力データＭＤＡ、ＭＤＢなどのデータが読み出される。このように本実施例ではいわゆるハーバードアーキテクチャのバス構成を採用している。

【００７７】ＰＡ＿ＢＵＳ、ＰＢ＿ＢＵＳ、ＷＷ＿ＢＵＳ、ＸＡ＿ＢＵＳは内部バスであり、ＡＵＸ＿ＢＵＳは制御回路１０２と積和演算回路１０４との間でデータのやり取りを行うためのバスである。ＩＡ、ＤＡは、各々、制御回路１０２（ＣＰＵ）からＩ＿ＡＤＤＲ＿ＢＵＳ、Ｄ＿ＡＤＤＲ＿ＢＵＳにアドレスを出力するためのものである。ＤＩＮは、Ｄ＿ＤＡＴＡ＿ＢＵＳからのデータを制御回路１０２に入力するためのものであり、ＤＯＵＴは、制御回路１０２からのデータをＤ＿ＤＡＴＡ＿ＢＵＳに出力するためのものである。

【００７８】命令デコーダ１４０は、Ｉ＿ＤＡＴＡ＿ＢＵＳから入力された命令を受け付けると共に解析し、命令の実行に必要な種々の制御信号を出力する。例えば命令に応じた種々の指示を、即値生成器１４２を介して制御回路１０２の各部に与える。また割り込みコントロー

ラ１３０（図１参照）からの割り込みを受け付けた場合には、割り込みハンドラを起動するＴＲＡＰ　ＶＥＣＴＯＲをＤ＿ＡＤＤＲ＿ＢＵＳに出力すると共に、ＴＲＡＰ信号をアクティブ（＝１）にして割り込みが発生したことを積和演算回路１０４に伝える。また積和演算命令を受け付けた場合には、ｍａｃ信号をアクティブにして積和演算命令が発行されたことを積和演算回路１０４に伝える。

【００７９】即値生成器１４２は、命令に含まれる即値に基づき、命令の実行時に使用する３２ビットの即値データを生成したり、各命令の実行に必要な０、±１、±２、±４のconstantデータを生成したりする。ＰＣインクリメンタ１１８は、１つの命令を実行する毎にＰＣ１２０の値をインクリメントする処理を行う。アドレス加算器１４４は、各種レジスタに格納されている情報や即値生成器１４２で生成される即値データを用いて加算処理を行い、メモリ１１０からの読み出し処理に必要なアドレスを生成する。

【００８０】汎用レジスタ１０３は１６本の３２ビットのレジスタＲ０～Ｒ１５を含んでいる。ＳＰ１４６は、スタックポインタ専用の３２ビットのレジスタであり、スタックの先頭番地を指すスタックポインタを格納する。ＰＳＲ（プロセッサステータスレジスタ）１４８は、各種のフラグを格納する３２ビットのレジスタである。　ＡＬＵ１３２は、算術演算や論理演算を行うものであり、本実施例では実行回数のデクリメント処理も行う。ゼロディテクタ１３４は、ＡＬＵ１３２の演算結果が零の場合にＡＬＵ＿ｚｅｒｏをアクティブ（＝１）にする。これによりＰＳＲ１４８にゼロフラグがセットされると共に、実行回数が零になったことが積和演算回路１０４に伝えられる。バスマルチプレクサ１２１は、ＰＡ＿ＢＵＳ、ＰＢ＿ＢＵＳ、ＷＷ＿ＢＵＳのいずれか１つを選択してＡＵＸ＿ＢＵＳに接続するためのものである。バスマルチプレクサ１２１はＴＥＭＰｍレジスタ１２２、ＴＥＭＰｎレジスタ１２４を含み、第１、第２の積和入力データＭＤＡ、ＭＤＢが両方とも揃った時にこれらのデータを積和演算回路１０４に出力する。

【００８１】積和演算回路１０４は、ステートマシーン１５０を含んでいる。このステートマシーン１５０は、ＡＬＵ＿ｚｅｒｏ、ｔｒａｐ、ｍａｃなどの各種の信号に基づいて積和演算回路１０４の状態を制御する。

【００８２】さて図１３のタイミングチャートの中のＭＡＣステート（ＭＡＣ０～ＭＡＣ８）は積和演算回路１０４（ステートマシーン１５０）の状態を表すものであり、図１４（Ａ）にその状態遷移図を示す。ここで状態遷移図の中の各信号の意味は次の通りである。

【００８３】　ｍａｃ
積和演算命令を命令デコーダ１４０が受け付けた時に１（アクティブ）になる信号である。

【００８４】　ｍａｃ＿ｅｎｄ
積和演算命令の終了条件が成立すると１になる信号であり、具体的にはｍａｃ＿ｚｅｒｏ又はｍａｃ＿ｔｒａｐが１になると１になる信号である。

【００８５】　ｍａｃ＿ｚｅｒｏ
積和演算の実行回数が零になった時に１になる信号である。ここで図１４（Ｂ）に示すように、ｍａｃ＿ｚｅｒｏは、マイクロコンピュータがリセットされた場合或いはＭＡＣステートがＭＡＣ８又はＭＡＣ９になった場合に０になる。またＭＡＣステートがＭＡＣ３、ＭＡＣ５又はＭＡＣ７の時にゼロディテクタ１３４からのＡＬＵ＿ｚｅｒｏ信号が１になると１になる。

【００８６】　ｍａｃ＿ｔｒａｐ
積和演算命令の実行中に割り込みが発生した場合に１になる信号である。ここで図１４（Ｂ）に示すように、ｍａｃ＿ｔｒａｐは、マイクロコンピュータがリセットされた場合或いはＭＡＣステートがＭＡＣ８又はＭＡＣ９になった場合に０になる。またＭＡＣステートがＭＡＣ５又はＭＡＣ７の時に命令デコーダ１４０からのｔｒａｐ信号が１になると１になる。

【００８７】図１４（Ａ）に示すように、積和演算命令が発行されずｍａｃ＝０の場合には、ＭＡＣステートはＭＡＣ０にとどまる。一方、ｍａｃ＝１になるとＭＡＣ１に移行する。ＭＡＣ１からＭＡＣ２、ＭＡＣ２からＭＡＣ３へはクロックに同期して無条件（ＵＣＴ）に移行する。

【００８８】ＭＡＣ３で、ｍａｃ＿ｅｎｄが１の場合にはＭＡＣ９に移行すると共に、ｍａｃ＿ｅｎｄが０にリセットされる（図１４（Ｂ）参照）。ＭＡＣ９に移行した後、ｍａｃ＝１ならＭＡＣ１に戻り、ｍａｃ＝０ならＭＡＣ０に戻る。一方、ｍａｃ＿ｅｎｄが０の場合にはＭＡＣ３からＭＡＣ４に移行する。

【００８９】ＭＡＣ４からＭＡＣ５へはクロックに同期して無条件に移行する。この際、実行回数がデクリメントされるため（図１３のＥ２２参照）、ｍａｃ＿ｚｅｒｏが１になる可能性がある。そこでＭＡＣ５で、ｍａｃ＿ｅｎｄが１か否かを判断し、１の場合にはＭＡＣ８に移行し、ＭＡＣ８からＭＡＣ０又はＭＡＣ１に戻る。一方、ｍａｃ＿ｅｎｄが０の場合は、ＭＡＣ５からＭＡＣ６に移行する。

【００９０】ＭＡＣ６からＭＡＣ７へはクロックに同期して無条件に移行する。この際、実行回数がデクリメントされるため（図１３のＥ２４、Ｅ２６参照）、ｍａｃ＿ｚｅｒｏが１になる可能性がある。そこでＭＡＣ７で、ｍａｃ＿ｅｎｄが１か否かを判断し、１の場合にはＭＡＣ８に移行し、０の場合はＭＡＣ６に戻る。

【００９１】例えば積和演算の実行回数が０に設定されていた場合には、ＭＡＣステートは、まずＭＡＣ０、ＭＡＣ１、ＭＡＣ２、ＭＡＣ３と変化する。そしてｍａｃ＿ｅｎｄ＝１（ｍａｃ＿ｚｅｒｏ＝１）となっているため、ＭＡＣ３からＭＡＣ９、ＭＡＣ０（又はＭＡＣ１）

と変化する。

【００９２】実行回数が１に設定されていた場合には、ＭＡＣステートは、まずＭＡＣ０、ＭＡＣ１、ＭＡＣ２、ＭＡＣ３、ＭＡＣ４と変化する。そしてＭＡＣ４からＭＡＣ５への移行の際に実行回数がデクリメントされるためｍａｃ＿ｅｎｄ＝１になる。この結果、ＭＡＣステートはＭＡＣ４からＭＡＣ５、ＭＡＣ８、ＭＡＣ０（又はＭＡＣ１）と変化することになる。

【００９３】実行回数が２に設定されていた場合には、ＭＡＣステートは、ＭＡＣ０、ＭＡＣ１、ＭＡＣ２、ＭＡＣ３、ＭＡＣ４、ＭＡＣ５、ＭＡＣ６、ＭＡＣ７、ＭＡＣ８、ＭＡＣ０（又はＭＡＣ１）と変化する。即ちこの場合には、ＭＡＣ４からＭＡＣ５、ＭＡＣ６からＭＡＣ７の間で実行回数がデクリメントされて零になる。なお実行回数が３以上の場合には、ＭＡＣ６からＭＡＣ７に移行しＭＡＣ６に戻る動作を実行回数が零になるまで繰り返すことになる。

【００９４】割り込み要求がなされた場合には、ＭＡＣ５又はＭＡＣ７まで状態が進んだ所で初めてｍａｃ＿ｔｒａｐ＝１（ｍａｃ＿ｅｎｄ＝１）か否かが判断され、ＭＡＣ８に移行することになる。

【００９５】本実施例のステートマシーン１５０の１つの特徴は、所望の回数の積和演算を完了した場合（ｍａｃ＿ｚｅｒｏ＝１）又は割り込み要求がなされた場合（ｍａｃ＿ｔｒａｐ＝１）にアクティブになるｍａｃ＿ｅｎｄ信号に基づいて、ＭＡＣステートを初期ステートＭＡＣ０（又はＭＡＣ１）に戻す点にある。このようにすることで、所望の回数の積和演算を完了した場合に用いる状態遷移を利用して、割り込み要求がなされた場合に行う状態遷移も実現することが可能となる。これによりステートマシーン１５０の構成の簡素化を図ることができる。

【００９６】次に図１３を用いて本実施例の動作を説明する。図１３は、実行回数が３に設定されている場合のタイミングチャートである。従って、この場合には、ＭＡＣステートは、ＭＡＣ０～ＭＡＣ６、ＭＡＣ７、ＭＡＣ６、ＭＡＣ７、ＭＡＣ８、ＭＡＣ０と変化することになる。また本実施例では汎用レジスタの中のＲ１３が実行回数用のレジスタになっており、実行回数３が設定されている（図１３のＥ０参照）。またＲ１４、Ｒ１５が第１、第２の積和入力データＭＤＡ、ＭＤＢ用のレジスタになっており、これらのレジスタには、各々、ＭＤＡ、ＭＤＢを格納するメモリ領域の先頭アドレス１１０ｈ、２３０ｈが格納されている（Ｅ１、Ｅ２参照）。

【００９７】図１２の命令デコーダ１４０が積和演算命令を受け付けるとｍａｃ＝１になり、ＭＡＣステートがＭＡＣ０からＭＡＣ１に移行する。

【００９８】次に、Ｒ１３に格納される実行回数がＰＢバスを介してＡＬＵ１３２に出力される（Ｅ３）。ＡＬＵ１３２は実行回数に０を加算する（Ｅ４）。ここで０を加算するのは、最初に設定された実行回数が０か否かを調べるためである。０の場合には、ＡＬＵ＿ｚｅｒｏが１になり積和演算命令の実行が終了する（図１４（Ａ）のＭＡＣ３、ＭＡＣ９参照）。

【００９９】次に、Ｒ１４に格納されるアドレス１１０ｈがＸＡ＿ＢＵＳを介してＤ＿ＡＤＤＲ＿ＢＵＳに出力される（Ｅ５、Ｅ６）。そしてこのアドレスに基づき第１の積和入力データＭＤＡ（１１０ｈ）がメモリ１１０から読み出される（Ｅ７）。同様にＲ１５に格納されるアドレス２３０ｈがＸＡ＿ＢＵＳを介してＤ＿ＡＤＤＲ＿ＢＵＳに出力され（Ｅ８、Ｅ９）、このアドレスに基づき第２の積和入力データＭＤＢ（２３０ｈ）がメモリ１１０から読み出される（Ｅ１０）。そしてこれらのＭＤＡ、ＭＤＢを乗算器１０５が乗算し（Ｅ１１）、乗算結果を加算器１０６-1が加算し（Ｅ１２）、加算結果をＡＬＲ１０７-1に格納する（Ｅ１３）。そして加算によりキャリー又はボローが生じた場合には、インクリメント・デクリメンタ１０６-2がインクリメント又はデクリメント処理を行い（Ｅ１４）、その結果をＡＨＲ１０７-2に格納する（Ｅ１５）。

【０１００】Ｒ１４、Ｒ１５に格納されるアドレス１１０ｈ、２３０ｈはＸＡ＿ＢＵＳを介してアドレス加算器１４４にも出力される（Ｅ５、Ｅ８）。アドレス加算器１４４は、これらのアドレスに＋２を加算し（Ｅ１６、Ｅ１７）、加算結果をＷＷ＿ＢＵＳを介してレジスタＲ１４、Ｒ１５に戻す（Ｅ１８、Ｅ１９）。これによりＲ１４、Ｒ１５に格納されるアドレスが１１２ｈ、２３２ｈに変化し（Ｅ２０、Ｅ２１）、次の積和入力データＭＤＡ（１１２ｈ）、ＭＤＢ（２３２ｈ）を読み出すことが可能になる。

【０１０１】ＡＬＵ１３２は、ＭＡＣ４において実行回数を３から２にデクリメントする（Ｅ２２）。そしてデクリメントされた実行回数がＰＢ＿ＢＵＳに出力され、ＰＢ＿ＢＵＳからＡＬＵ１３２の入力に戻される（Ｅ２３）。次にＡＬＵ１３２は、実行回数を２から１にデクリメントする（Ｅ２４）。そしてデクリメントされた実行回数がＡＬＵ１３２の入力に戻される（Ｅ２５）。次にＡＬＵ１３２は、実行回数を１から０にデクリメントする（Ｅ２６）。すると実行回数が０になったのでＡＬＵ＿ｚｅｒｏが１になる（Ｅ２７）。するとＭＡＣステートがＭＡＣ６からＭＡＣ７、ＭＡＣ８、ＭＡＣ０と変化し（Ｅ２８）、積和演算命令の実行が終了する。この際、デクリメントされて０になった実行回数はＷＷ＿ＢＵＳを介してＲ１３に格納されることになる（Ｅ２９、Ｅ３０）。

【０１０２】次に図１５のタイミングチャートを用いて、割り込みが発生した場合の本実施例の動作について説明する。図１５に示すように、例えばＭＡＣステートがＭＡＣ３の時に割り込みが発生しｔｒａｐ信号が１になった場合（図１５のＦ１）を考える。この場合、本実

施例では、次のＭＡＣステートであるＭＡＣ４では、割り込みがなかった場合と同様の処理が行われる。そしてＭＡＣステートがＭＡＣ５になった時に初めて、ＭＡＣ５からＭＡＣ８、ＭＡＣ０と変化する処理を行う（Ｆ２）。

【０１０３】即ち本実施例では、レジスタＲ１４、Ｒ１５の内容が、割り込み処理終了後の積和演算の継続実行の際に使用される内容に変化した後に（Ｆ３、Ｆ４、Ｆ５、Ｆ６）、ＭＡＣステートが初期ステートＭＡＣ０（又はＭＡＣ１）に戻る。このようにすることで、割り込み処理の終了後に、１１２ｈ、２３２ｈのアドレスにある積和入力データＭＤＡ、ＭＤＢに基づいて積和演算を適切に継続実行することが可能となる。

【０１０４】また本実施例では、積和演算の実行回数がデクリメントした後に（Ｆ７）、ＭＡＣステートが初期ステートに戻る。従って、Ｒ１３には、デクリメント後の実行回数２が格納されることになり（Ｆ８、Ｆ９）、割り込み処理の終了後に、残りの２回の積和演算を継続実行することが可能となる。

【０１０５】以上のように処理することで、前述の実施例１、２、４で説明した種々の処理を実現できる。なお実施例３の処理を実現するためには、積和入力データＭＤＡ、ＭＤＢのメモリからの読み出しを、１回のメモリアクセス（１クロック）で行うようにすればよい。

【０１０６】（実施例６）実施例６は、本発明が適用されるマイクロコンピュータの詳細例について説明する実施例である。

【０１０７】図１６に示すように実施例６のマイクロコンピュータ７００は、３２ビットマイクロコンピュータであり、ＣＰＵ（制御回路、積和演算回路、ＡＬＵ）７１０、ＲＯＭ７２０、ＲＡＭ７３０、高周波発振回路９１０、低周波発振回路９２０、リセット回路９３０、プリスケーラ９４０、１６ビットプログラマブルタイマ９５０や８ビットプログラマブルタイマ９６０やクロックタイマ９７０などのタイマ回路、インテリジェントＤＭＡ９８０や高速ＤＭＡ９９０などのデータ転送制御回路、割り込みコントローラ８００、シリアルインターフェース８１０、ＢＣＵ（バスコントロールユニット）７４０、Ａ／Ｄ変換器８３０やＤ／Ａ変換器８４０などのアナログインターフェース回路、入力ポート８５０や出力ポート８６０やＩ／Ｏポート８７０などのＩ／Ｏ回路、及びそれらを接続する各種バス７５０、７６０、各種ピン８９０を含む。

【０１０８】１チップの半導体基板上に形成されるこのマイクロコンピュータ７００は、３２ビットのデータを処理できるＲＩＳＣ方式のマイクロコンピュータである。そしてパイプライン方式及びロード・ストア方式のアーキテクチャーを採用し、ほとんど全ての命令を１クロックの期間で実行する。全ての命令は１６ビットの固定長で記述されており、これにより極めて小さい命令コードサイズを実現している。

【０１０９】そして、実施例１～５で説明したように、ＣＰＵ７１０は、１つの積和演算命令で複数回の積和演算を実行できるようになっている。このため、このマイクロコンピュータ７００は、これまでＤＳＰ、画像処理専用ＩＣ、音処理専用ＩＣなどが行っていた処理を代行することができ、このマイクロコンピュータ７００が組み込まれる電子機器の低コスト化、小型化を図ることが可能となる。

【０１１０】（実施例７）実施例７は、実施例１～６で説明したマイクロコンピュータを含む電子機器に関する実施例である。

【０１１１】例えば図１７（Ａ）に電子機器の１つであるカーナビゲーションシステムの内部ブロック図を示し、図１８（Ａ）にその外観図を示す。カーナビゲーションシステムの操作はリモコン５１０を用いて行われ、ＧＰＳやジャイロからの情報に基づいて位置検出部５２０が車の位置を検出する。地図などの情報はＣＤＲＯＭ５３０（情報記憶媒体）に格納されている。画像メモリ５４０は画像処理の際の作業領域になるメモリであり、生成された画像は画像出力部５５０を用いてドライバーに表示される。マイクロコンピュータ５００は、リモコン５１０、位置検出部５２０、ＣＤＲＯＭ５３０などのデータ入力源からデータを入力し、種々の処理を行い、処理後のデータを画像出力部５５０などの出力装置を用いて出力する。

【０１１２】これまでのカーナビゲーションシステムでは、画像処理（グラフィック処理）は、ＤＳＰや専用の画像処理ＩＣが行っていた。このため、例えばＣＩＳＣ型のマイクロコンピュータとＤＳＰというように電子機器内に２つのプロセッサが存在することになり、システムが複雑化していた。実施例１～６で説明したマイクロコンピュータを採用すれば、複数回の積和演算命令の実行を効率よく行うことができるため、ＤＳＰ等を用いることなく、カーナビゲーションシステムが必要とする画像処理を実現することが可能となる。

【０１１３】図１７（Ｂ）に電子機器の１つであるゲーム装置の内部ブロック図を示し、図１８（Ｂ）にその外観図を示す。このゲーム装置では、ゲームコントローラ５６０からのプレーヤの操作情報、ＣＤＲＯＭ５７０からのゲームプログラム、ＩＣカード５８０からのプレーヤ情報等に基づいて、画像メモリ５９０を作業領域としてゲーム画像やゲーム音を生成し、画像出力部６１０、音出力部６００を用いて出力する。マイクロコンピュータ５００は、実施例１～６で説明した積和演算機能を用いて、座標変換、透視変換、クリッピングなどの３次元画像処理や、音圧縮、音伸長などの音処理を行うことになる。

【０１１４】図１７（Ｃ）に電子機器の１つであるプリンタの内部ブロック図を示し、図１８（Ｃ）にその外観

図を示す。このプリンタでは、操作パネル６２０からの操作情報、コードメモリ６３０及びフォントメモリ６４０から文字情報に基づいて、ビットマップメモリ６５０を作業領域として、印刷画像を生成し、プリント出力部６６０を用いて出力する。またプリンタの状態やモードを表示パネル６７０を用いてユーザに伝える。マイクロコンピュータ５００は、実施例１～６で説明した積和演算機能を用いて、直線や円弧の描画、画像の拡大、縮小などの処理を行うことになる。

【０１１５】なお本発明のマイクロコンピュータを適用できる電子機器としては、上記以外にも例えば、携帯電話（セルラーフォン）、ＰＨＳ、ページャ、オーディオ機器、電子手帳、電子卓上計算機、ＰＯＳ端末、タッチパネルを備えた装置、プロジェクタ、ワードプロセッサ、パーソナルコンピュータ、テレビ、ビューファインダ型又はモニタ直視型のビデオテープレコーダなど種々のものを考えることができる。

【０１１６】なお、本発明は上記実施例１～７に限定されるものではなく、本発明の要旨の範囲内で種々の変形実施が可能である。

【０１１７】例えば積和演算命令による積和演算の実行回数の特定手法は、上記実施例で説明したものに限らず、種々の変形実施が可能である。

【０１１８】また本発明の情報処理回路は、マイクロコンピュータ、特にＲＩＳＣ型のマイクロコンピュータに適用した場合に特に有効であるが、それ以外の用途も可能である。

【０１１９】また積和演算命令の記述構成も本実施例で説明したものに限られるものではなく、種々の変形実施が可能である。

【０１２０】

【図面の簡単な説明】

【図１】マイクロコンピュータの構成例を示すブロック図である。

【図２】実施例１の動作を説明するためのフローチャートである。

【図３】図３（Ａ）、（Ｂ）、（Ｃ）は、レジスタとメモリに格納されるデータの関係について説明するための図である。

【図４】比較例の動作を説明するためのフローチャートである。

【図５】図５（Ａ）、（Ｂ）は、比較例の問題点について説明するための図である。

【図６】図６（Ａ）、（Ｂ）、（Ｃ）は、積和演算命令の種々の実施形態について説明するための図である。

【図７】実施例２の動作を説明するためのフローチャートである。

【図８】図８（Ａ）、（Ｂ）は、レジスタとメモリに格納されるデータの関係について説明するための図である。

【図９】実施例３の動作を説明するためのフローチャートである。

【図１０】図１０（Ａ）は、実施例３のメモリへのデータ格納手法について説明するための図であり、図１０（Ｂ）は比較例のタイミングチャートの例であり、図１０（Ｃ）は、実施例３のタイミングチャートの例である。

【図１１】図１１（Ａ）は実施例４の構成例を示すブロック図であり、図１１（Ｂ）はそのタイミングチャートの例である。

【図１２】実施例５の構成例を示すブロック図である。

【図１３】実施例５のタイミングチャートの例である。

【図１４】図１４（Ａ）、（Ｂ）は、ステートマシーンについて説明するための図である。

【図１５】割り込み発生時のタイミングチャートの例である。

【図１６】実施例６のマイクロコンピュータの構成例である。

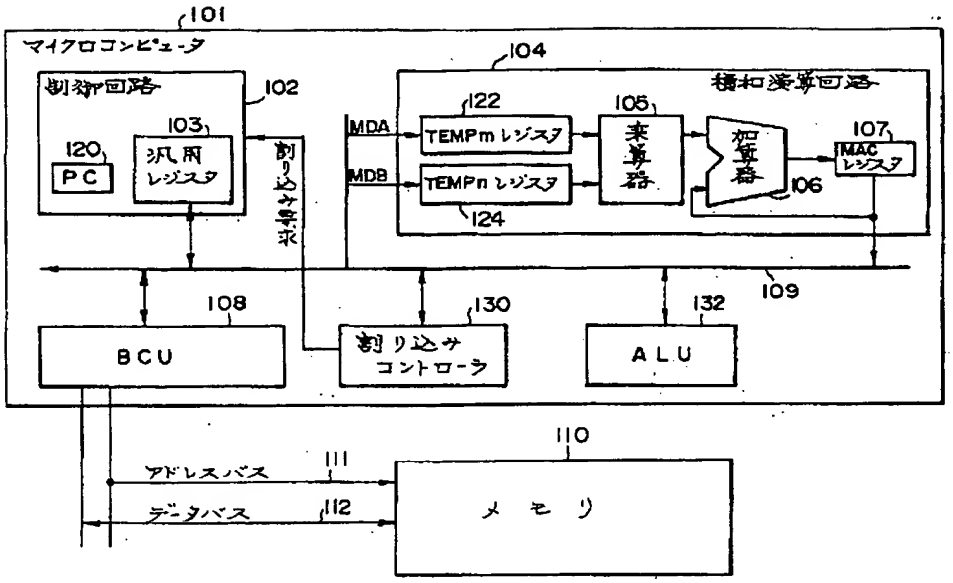
【図１７】図１７（Ａ）、（Ｂ）、（Ｃ）は、種々の電子機器の内部ブロック図の例である。

【図１８】図１８（Ａ）、（Ｂ）、（Ｃ）は、種々の電子機器の外観図の例である。

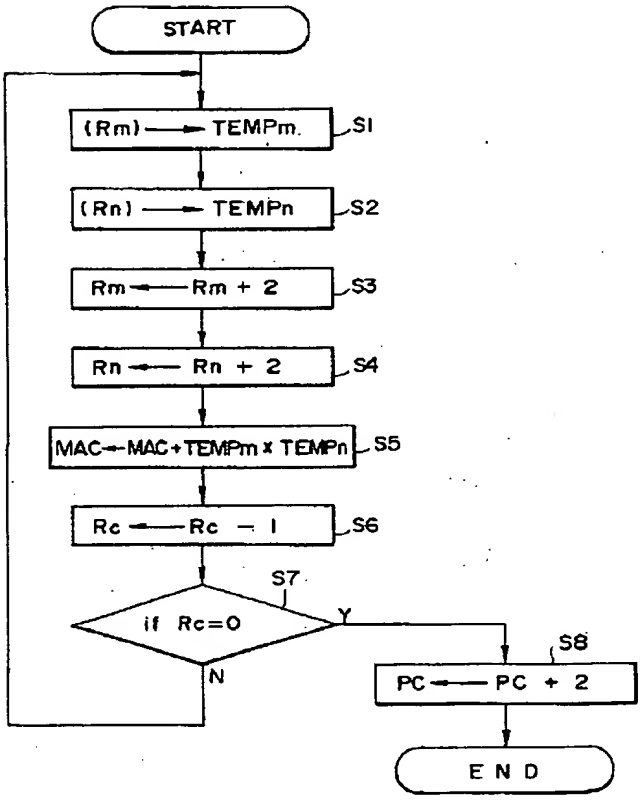
【符号の説明】

１０１　マイクロコンピュータ
１０２　制御回路
１０３　汎用レジスタ
１０４　積和演算回路
１０５　乗算器
１０６　加算器
１０７　ＭＡＣレジスタ（積和結果用レジスタ）
１０８　ＢＣＵ（バスコントロールユニット）
１０９　内部データバス
１１０　メモリ
１１１　外部アドレスバス
１１２　外部データバス
１２０　ＰＣ（プログラムカウンタ）
１２２　ＴＥＭＰｍレジスタ
１２４　ＴＥＭＰｎレジスタ
１３０　割り込みコントローラ
１３２　ＡＬＵ

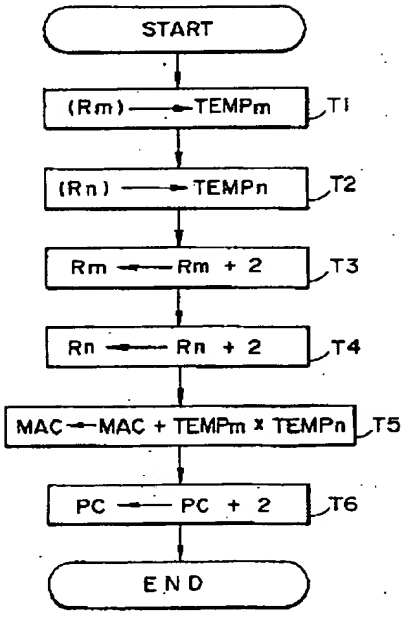
【図１】


101
マイクロコンピュータ
制御回路
102
103
汎用レジスタ
120
PC
割り込み要求
104
積和演算回路
122
TEMPm レジスタ
105
乗算器
107
MAC レジスタ
MDA
MDB
TEMPn レジスタ
124
106
109
108
BCU
130
割り込みコントローラ
132
ALU
110
アドレスバス
111
データバス
112
メモリ

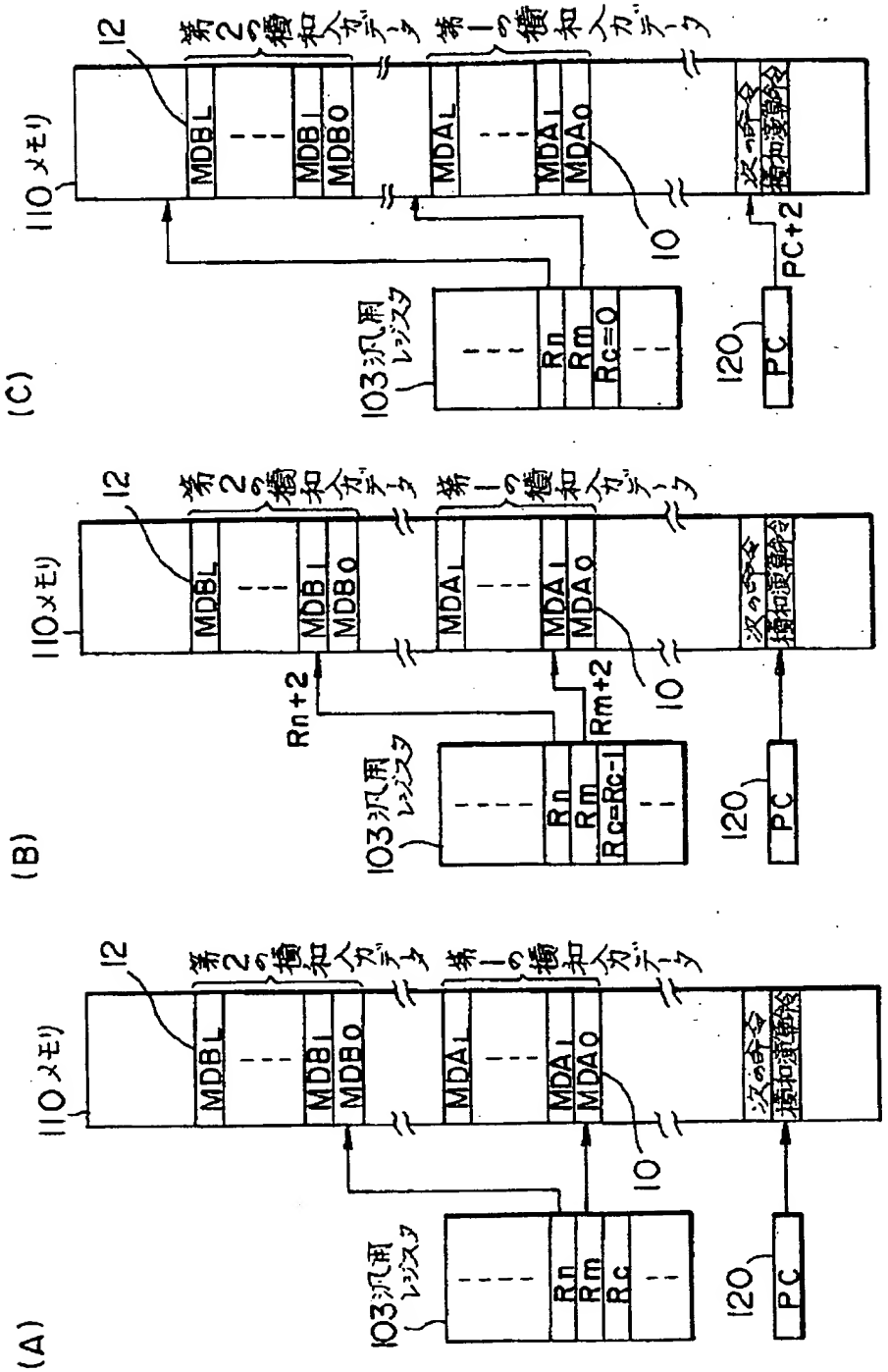
【図２】


START
(Rm) → TEMPm
S1
(Rn) → TEMPn
S2
Rm ← Rm + 2
S3
Rn ← Rn + 2
S4
MAC ← MAC + TEMPm × TEMPn
S5
Rc ← Rc − 1
S6
S7
if Rc=0
Y
N
S8
PC ← PC + 2
END

【図４】


START
(Rm) → TEMPm
T1
(Rn) → TEMPn
T2
Rm ← Rm + 2
T3
Rn ← Rn + 2
T4
MAC ← MAC + TEMPm × TEMPn
T5
PC ← PC + 2
T6
END

【図３】

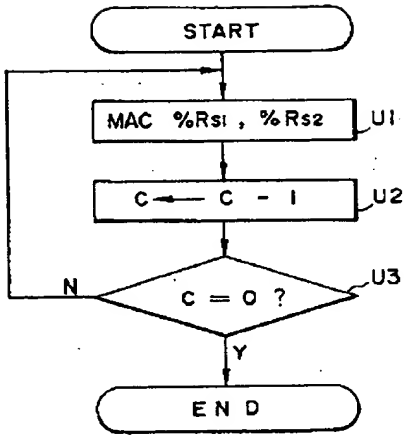

(A)
(B)
(C)
110メモリ
12
第2の積和入力データ
第1の積和入力データ
MDBL
MDBI
MDB0
MDAL
MDAI
MDA0
次のデータ
積和演算命令
10
103汎用レジスタ
Rn
Rm
Rc
Rc=0
Rc=Rc-1
Rn+2
Rm+2
120
PC
PC+2

【図５】

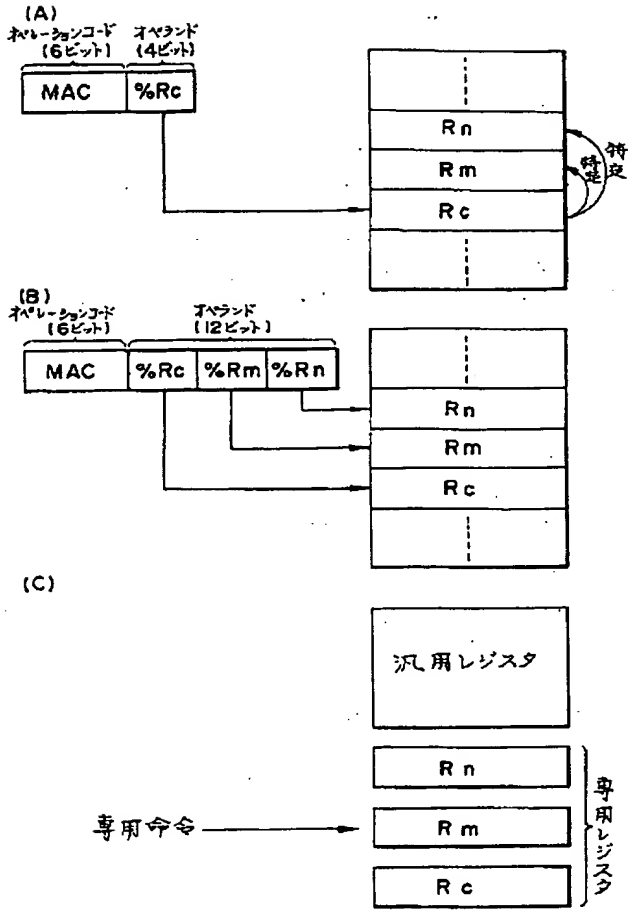
(A)

```
MAC   %Rs1 , %Rs2
MAC   %Rs1 , %Rs2
MAC   %Rs1 , %Rs2
MAC   %Rs1 , %Rs2
  ⋮
```

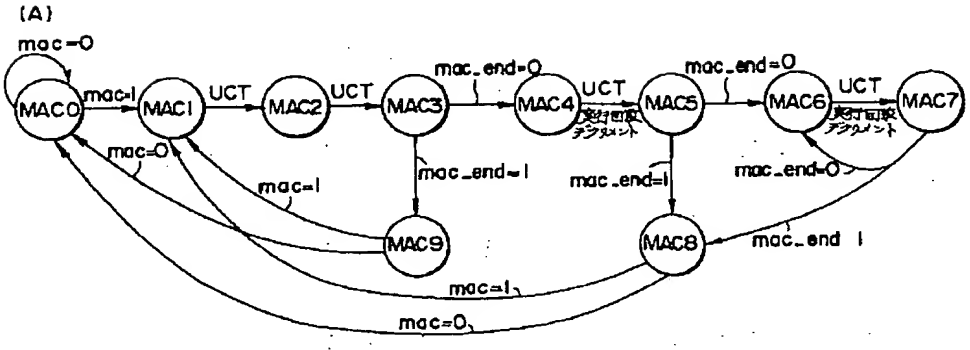
(B)



【図６】



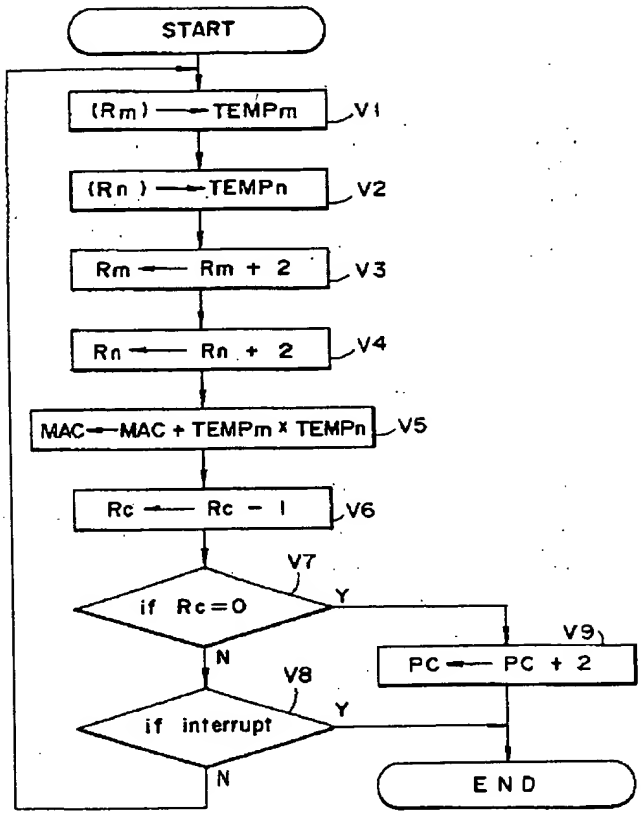
【図１４】



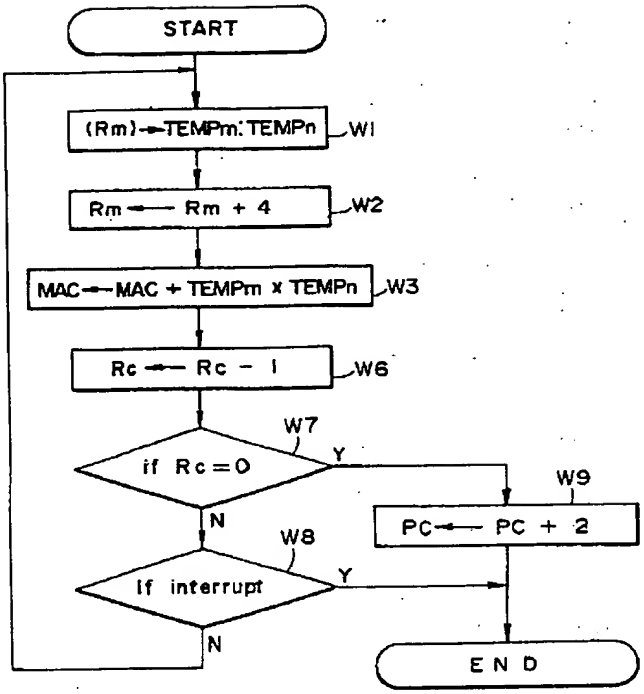
(B)

		インアクティブ(=0)条件	アクティブ(=1)条件
mac_end (mac_zero or mac_trap)	mac_zero	リセット or MAC8 or MAC9	ALU_zero AND (MAC3 or MAC5 or MAC7)
	mac_trap	リセット or MAC8 or MAC9	trap AND (MAC5 or MAC7)

【図７】

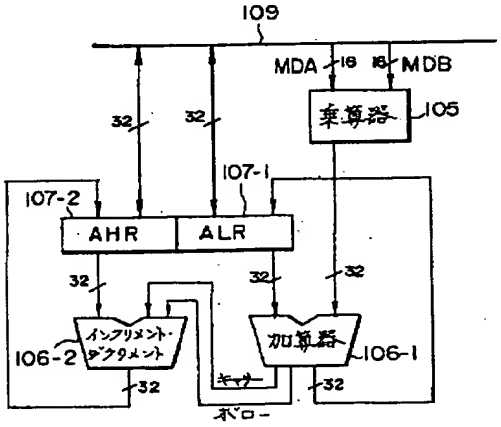

START
(Rm) → TEMPm
V1
(Rn) → TEMPn
V2
Rm ← Rm + 2
V3
Rn ← Rn + 2
V4
MAC ← MAC + TEMPm × TEMPn
V5
Rc ← Rc − 1
V6
V7
if Rc=0
Y
N
V9
PC ← PC + 2
V8
if interrupt
Y
N
END

【図９】

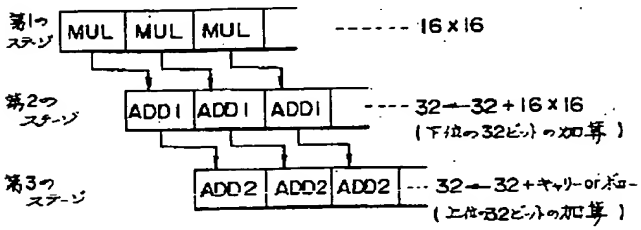

START
(Rm)→TEMPm; TEMPn
W1
Rm ← Rm + 4
W2
MAC ← MAC + TEMPm × TEMPn
W3
Rc ← Rc − 1
W6
W7
if Rc=0
Y
N
W9
PC ← PC + 2
W8
if interrupt
Y
N
END

【図１１】

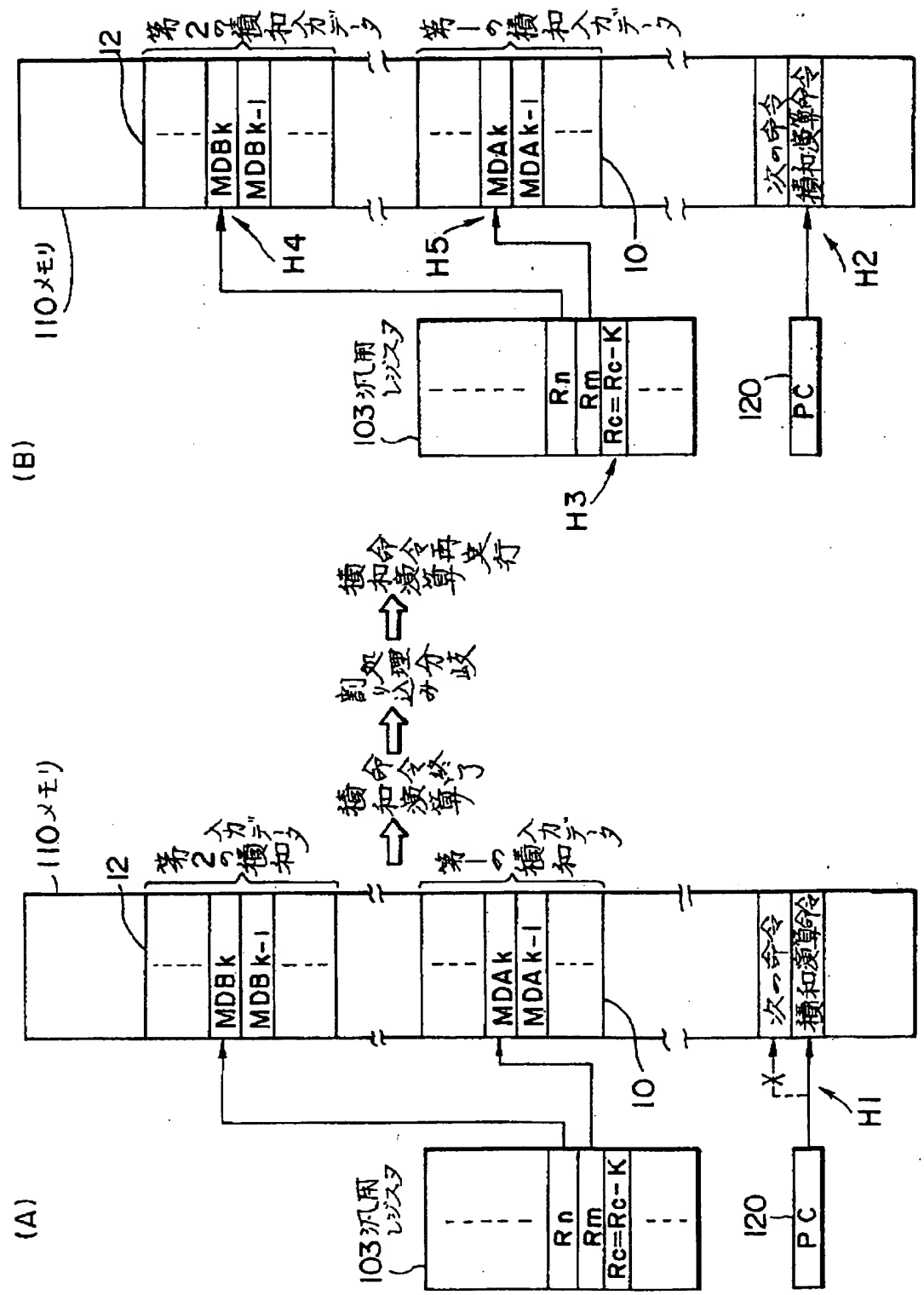
(A)


109
MDA
16
16
MDB
32
32
乗算器
105
107-1
107-2
AHR
ALR
32
32
32
インクリメント・デクリメント
106-2
加算器
106-1
キャリー
32
32
ボロー

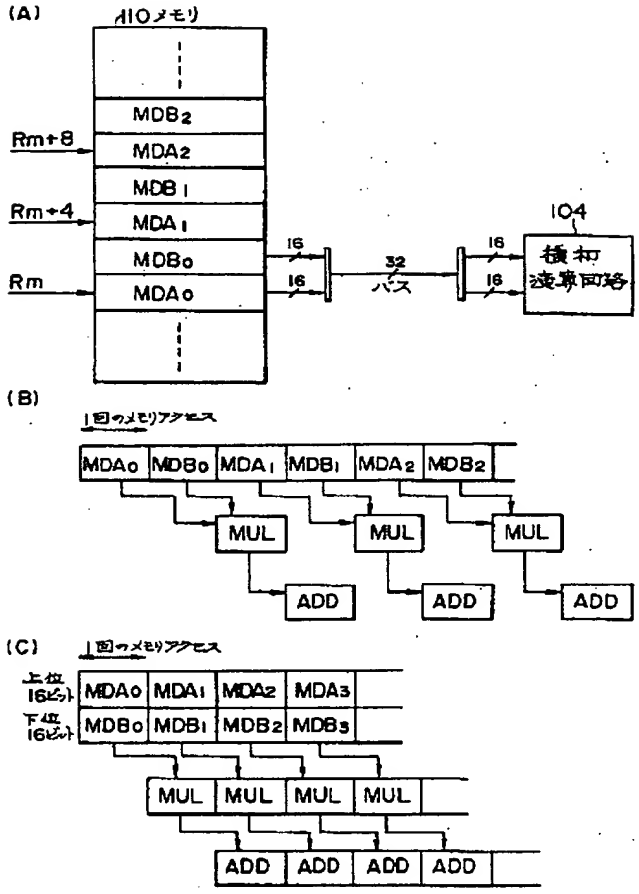
(B)


第1のステージ
MUL
MUL
MUL
------ 16×16
第2のステージ
ADD1
ADD1
ADD1
---- 32←32+16×16
(下位の32ビットの加算)
第3のステージ
ADD2
ADD2
ADD2
--- 32←32+キャリーorボロー
(上位32ビットの加算)

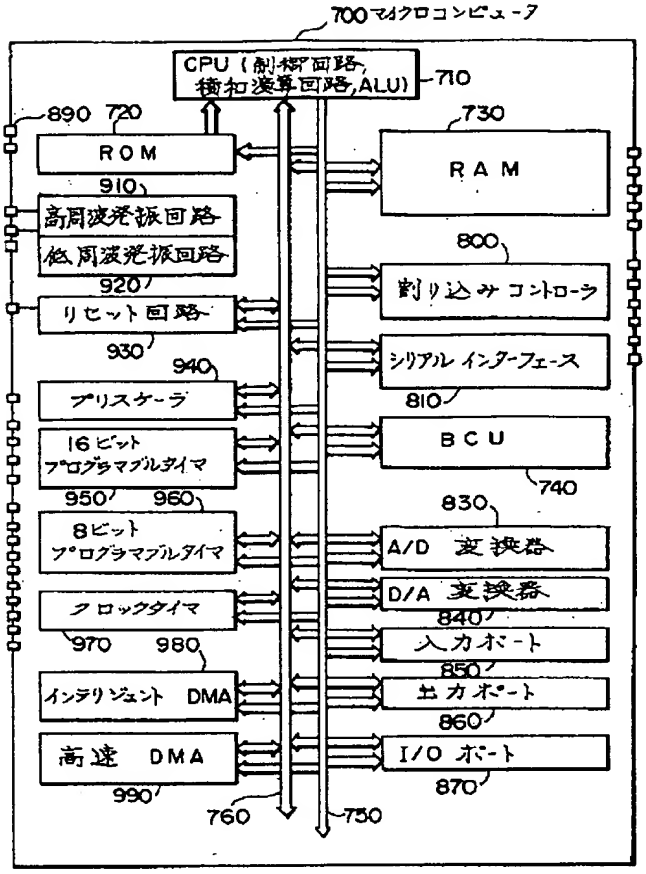
【図８】


(A)
110メモリ
12
第2の積和入力データ
第1の積和入力データ
MDBk
MDBk-1
MDAk
MDAk-1
次の命令
積和演算命令
10
H1
103汎用レジスタ
Rn
Rm
Rc=Rc-K
120
PC
積和命令実行終了
割り込み処理分岐
積和命令再実行
(B)
110メモリ
12
第2の積和入力データ
第1の積和入力データ
MDBk
MDBk-1
MDAk
MDAk-1
次の命令
積和演算命令
10
H4
H5
H2
103汎用レジスタ
Rn
Rm
Rc=Rc-K
120
PC
H3

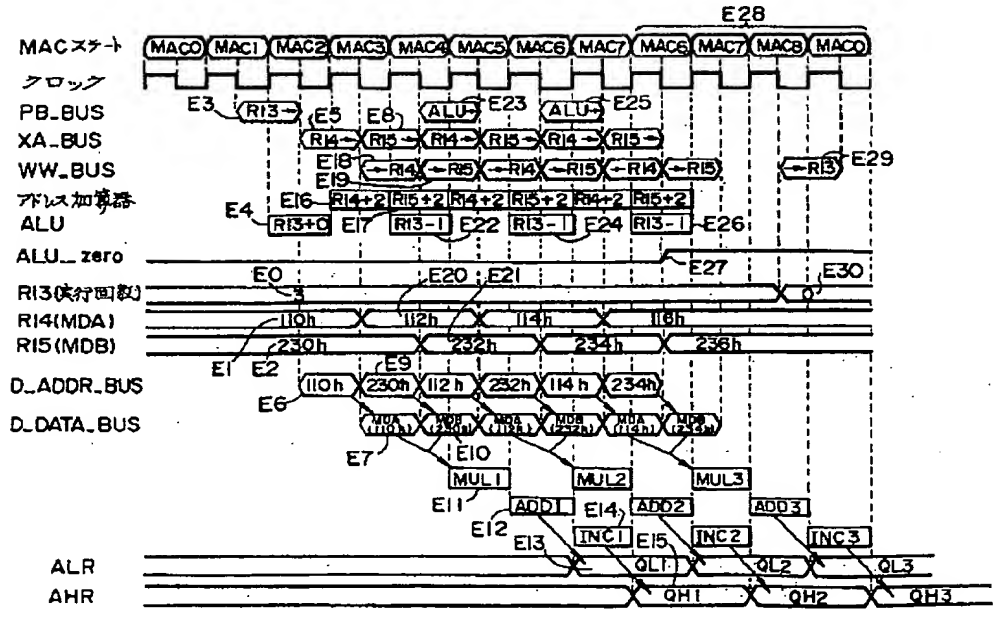
【図１０】


(A)
110メモリ
MDB2
MDA2
MDB1
MDA1
MDB0
MDA0
Rm+8
Rm+4
Rm
16
16
32
バス
16
16
104
積和演算回路
(B)
1回のメモリアクセス
MDA0 MDB0 MDA1 MDB1 MDA2 MDB2
MUL
ADD
(C)
1回のメモリアクセス
上位16ビット
下位16ビット
MDA0 MDA1 MDA2 MDA3
MDB0 MDB1 MDB2 MDB3
MUL
ADD

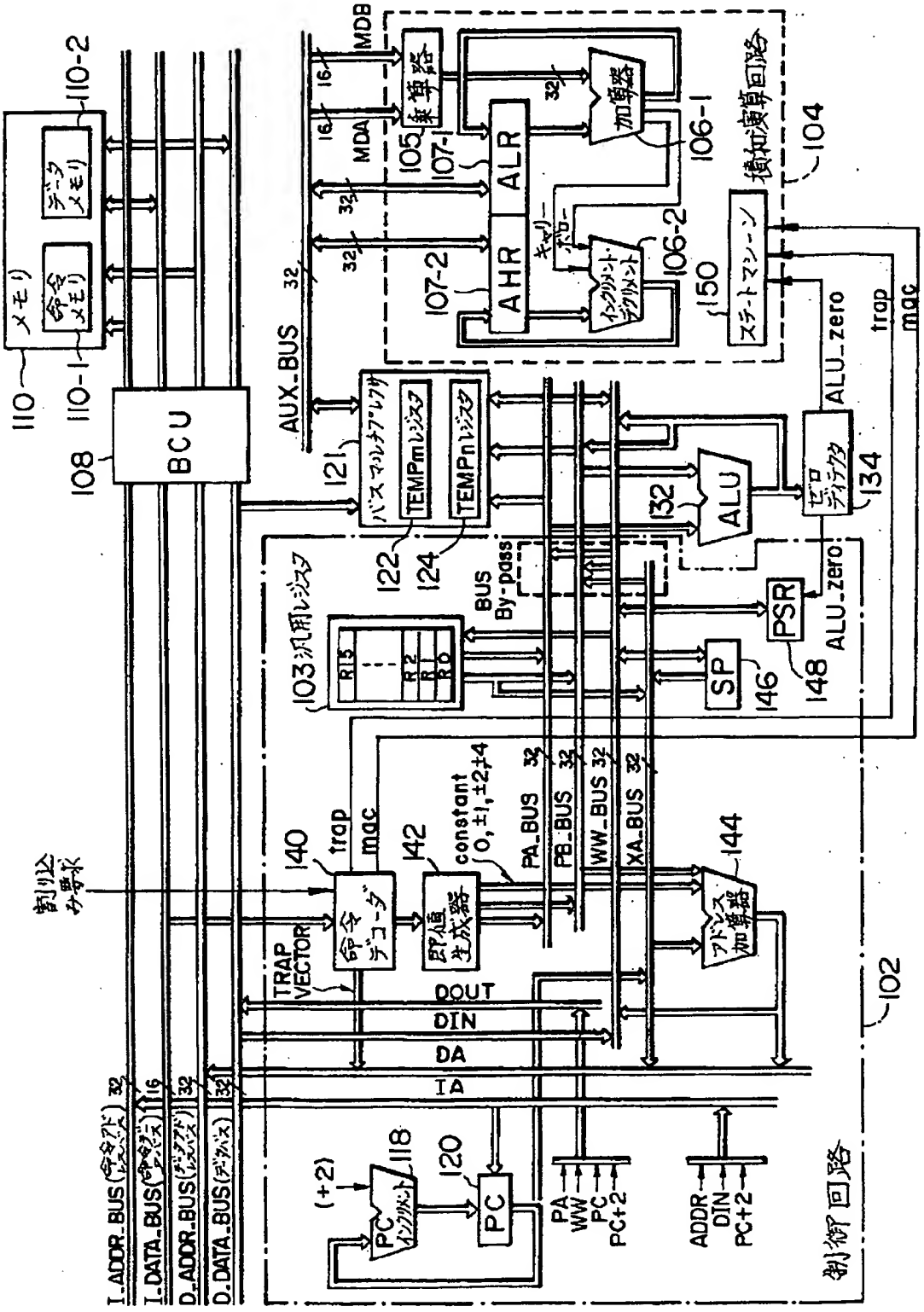
【図１６】


700マイクロコンピュータ
CPU（制御回路,積和演算回路,ALU）
710
890
720
ROM
910
高周波発振回路
低周波発振回路
920
リセット回路
930
940
プリスケーラ
16ビット プログラマブルタイマ
950
960
8ビット プログラマブルタイマ
クロックタイマ
970
980
インテリジェント DMA
高速 DMA
990
760
750
730
RAM
800
割り込みコントローラ
シリアル インターフェース
810
BCU
740
830
A/D 変換器
D/A 変換器
840
入力ポート
850
出力ポート
860
I/O ポート
870

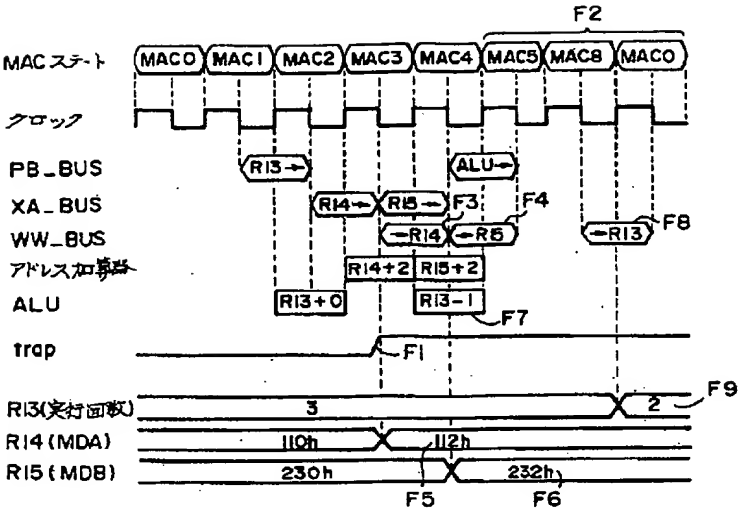
【図１３】


MACステート
クロック
PB_BUS
XA_BUS
WW_BUS
アドレス加算器
ALU
ALU_zero
RI3(実行回数)
RI4(MDA)
RI5(MDB)
D_ADDR_BUS
D_DATA_BUS
ALR
AHR
MAC0 MAC1 MAC2 MAC3 MAC4 MAC5 MAC6 MAC7 MAC6 MAC7 MAC8 MAC0
E28
E3 E5 E8 E23 E25 E29
E18 E19 E16 E4 E17 E22 E24 E26
E0 E20 E21 E27 E30
E1 E2 E9 E6 E7 E10 E11 E12 E13 E14 E15
110h 112h 114h 116h
230h 232h 234h 236h
MUL1 MUL2 MUL3
ADD1 ADD2 ADD3
INC1 INC2 INC3
QL1 QL2 QL3
QH1 QH2 QH3

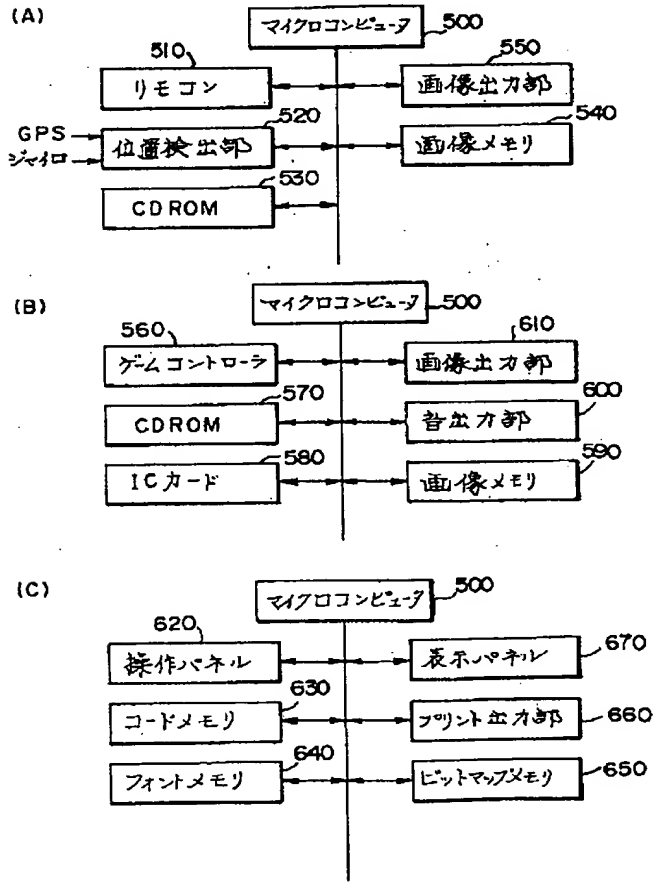
【図１２】


メモリ
命令メモリ
データメモリ
110
110-1
110-2
BCU
108
I_ADDR_BUS(命令アドレスバス) 32
I_DATA_BUS(命令データバス) 16
D_ADDR_BUS(データアドレスバス) 32
D_DATA_BUS(データバス) 32
MDB
16
MDA
105 乗算器
107-1
ALR
AHR
107-2
32
加算器
106-1
キャリーセーブ
インクリメント・デクリメント
106-2
積和演算回路
104
ステートマシーン
150
AUX_BUS
121 バスマルチプレクサ
TEMPmレジスタ
TEMPnレジスタ
122
124
ALU
132
ゼロディテクタ
134
ALU_zero
trap
mac
BUS By-pass
103 汎用レジスタ
R15
R2
R1
R0
PSR
148
SP
146
割り込み命令
140
trap
mac
命令デコーダ
142
即値生成器
constant 0,±1,±2,±4
PA_BUS 32
PB_BUS 32
WW_BUS 32
XA_BUS 32
144
アドレス加算器
TRAP VECTOR
DOUT
DIN
DA
IA
102
(+2)
PC インクリメント
118
120
PC
PA
WW
PC
PC+2
ADDR
DIN
PC+2
制御回路

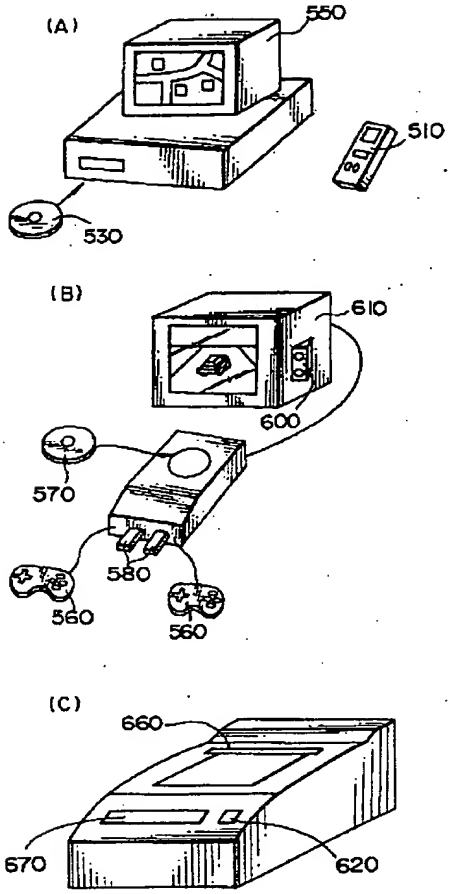
【図１５】


MACステート
クロック
PB_BUS
XA_BUS
WW_BUS
アドレス加算器
ALU
trap
R13(実行回数)
R14(MDA)
R15(MDB)
MAC0
MAC1
MAC2
MAC3
MAC4
MAC5
MAC8
MAC0
R13→
R14→
R15→
←R14
←R15
R14+2
R15+2
R13+0
R13-1
ALU→
←R13
F1
F2
F3
F4
F5
F6
F7
F8
F9
3
2
110h
112h
230h
232h

【図１７】


(A)
マイクロコンピュータ
500
510
リモコン
550
画像出力部
GPS
ジャイロ
520
位置検出部
540
画像メモリ
530
CD ROM
(B)
マイクロコンピュータ
500
560
ゲームコントローラ
610
画像出力部
570
CDROM
600
音出力部
580
ICカード
590
画像メモリ
(C)
マイクロコンピュータ
500
620
操作パネル
670
表示パネル
630
コードメモリ
660
プリント出力部
640
フォントメモリ
650
ビットマップメモリ

【図１８】


(A)
550
510
530
(B)
610
600
570
580
560
560
(C)
660
670
620